Haaga-Helia
ammattikorkeakoulu Oy

Software Engineerin työpäiväkirja – pala junior web-kehittäjän arkea

Arto Tukki

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2018

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

2.6.2018

Tekijä(t)
Arto Tukki

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Software Engineerin työpäiväkirja – pala junior web-kehittäjän arkea

Sivu- ja liitesivumäärä
60

Opinnäytetyön otsikko englanniksi
Work diary of a Software Engineer – a glimpse to the life of a junior web developer

Tässä päiväkirjamuotoisessa opinnäytetyössä seurataan junior tasoista web-kehittäjää ensimmäisessä ohjelmistokehitystyössä. Tarkoituksena tässä työssä on seurata opiskelijan tietojen ja taitojen kehitystä 8 viikon ajan sekä kuvata web-kehittäjän työtehtäviä, eteen tulevia ongelmia ja miten ne on ratkaistu.

Opinnäytetyö on tehty siten, että jokaiselle työpäivälle on asetettu tavoite, seurattu sen toteutumista ja lisäksi mahdollisesti analysoitu päivän tapahtumia. Lisäksi työssä on jokaisen työviikon jälkeen pohdittu menneen viikon tapahtumia ja niihin liittyviä aiheita ohjelmistokehittäjän uraan ja ohjelmistokehitykseen liittyen.

Tämä opinnäytetyö on toteutettu keskikokoisessa suomalaisessa ohjelmistoalan yrityksessä, joka myy muun muassa ohjelmistokehityskonsultointia asiakkaille. Opinnäytetyössä on kuvattu opiskelijan asiakkaalle tehtyä työtä sillä tarkkuudella, mitä salassapitosopimuksen puitteissa on pystytty.

Seurantajakson aikana huomattiin, että opiskelijan ohjelmistokehittäjän työssä tarvittava ammattitaito on kehittynyt paljon, kuinka erilaista ohjelmistokehitystyö on verrattuna harrasteluun ja minkälaista on työskennellä asiakkaan tiloissa konsulttina.

Asiasanat
Ohjelmistokehitys, verkko-ohjelmointi, tietotekniikka-ala, ohjelmointi, ohjelmistoala

Sisällys

1 Johdanto

Opinnäytetyö tehdään päiväkirjaopinnäytetyönä, mikä tarkoittaa, että raportointi tapahtuu päivittäin sen hetkisten työtehtävien kuvaamisella, sillä tarkkuudella, millä salassapitosopimus sen sallii, sekä viikoittaisella viikkoanalyysillä. Opinnäytetyö suoritetaan aikavälillä 9.4.2018 – 1.6.2018.

Työnantajani on keskikokoinen ohjelmistoalan yritys, jonka asiakkaita ovat monet pääkaupunkiseudun yritykset, mikä tarkoittaa sitä, että pääsääntöisesti työtehtävänäni on toimia ohjelmoijana / konsulttina eri asiakkailla. Yleensä olen mukana yhden asiakkaan projektissa kerrallaan. Yritys on perustettu vuonna 2005 ja sen kotipaikkana on Helsinki, mutta yrityksellä on toimintaa myös muun muassa Ruotsissa ja Tanskassa.

Työtehtävissäni tarvitaan monipuolista ohjelmointialan osaamista, mutta myös sosiaaliset taidot ovat äärimmäisen tärkeät, sillä työtehtävät suoritetaan lähes poikkeuksetta asiakkaan tiloissa, jolloin pitää kyetä kommunikoimaan erilaisten ihmisten kanssa. Lisäksi ongelmanratkaisutaidot ovat suuressa osassa päivittäisiä työtehtäviä, koska tarkoituksena on tarjota laadukas ratkaisu asiakkaan esittämään ongelmaan. Työtehtävässäni ei ole vaatimuksena IT-alan koulutusta, mutta se lasketaan eduksi. Työpäivät koostuvat pääosin ohjelmoinnista, projektiryhmän keskinäisestä vapaamuotoisesta kommunikoinnista sekä mahdollisista kokouksista, joiden määrä on hyvin riippuvainen meneillään olevasta projektista ja sen vetäjästä.

Projektiryhmään, missä olen tällä hetkellä mukana, kuuluu lisäkseni kolme kollegaani, joista yksi on mukana tässä projektissa vain osa-aikaisesti. Projektissa nimikkeeni on full-stack kehittäjä, joka tarkoittaa sitä, että työskentelen sekä käyttöliittymän että serveripuolen ohjelmoinnin parissa.

1.1 Työssäni tarvittava osaaminen ja keskeiset käsitteet:

Angular on Googlen kehittämä komponentteihin perustuva JavaScript ohjelmistokehys, joka nopeuttaa kehitystyötä tarjoamalla valmiita ratkaisuja, työkaluja ja komponentteja. Komponentit voivat olla niin sanotusti sukua toisilleen, mikä tarkoittaa, että jos komponentin sisälle laitetaan toinen komponentti, sukulaisuussuhde on silloin lapsi-vanhempi. Kansiorakenteessa samalla tasolla olevat komponentit ovat sisaruksia toisilleen. Angular tunnettiin ennen nimellä AngularJS, mutta nimeäminen muuttui toisen version myötä, sillä siihen tehtiin niin suuria muutoksia. Versiot Angular 2:sta (4, 5 ja 6) eteenpäin ovat sisältäneet lähinnä pieniä muutoksia. Angularin versiota 3 ei koskaan tullut, vaan Angular 2:sta

hypättiin versionumeroinnissa suoraan versioon numero neljä. Tämä johtui siitä, että Angularin käyttämä reititin (router) oli jo versiossa kolme ja haluttiin, että sekä ohjelmistokehys, että sen käyttämä reititin olisi samassa versionumerossa.

Backend on sekä sovelluksen varsinainen palvelin että sovelluksen palvelinpuolen ohjelmakoodi. Yleensä kaikki bisneslogiikka toteutetaan backendissä.

Bugi on ohjelmakoodissa esiintyvä virhe, joka yleensä aiheuttaa joko koko sivun, yhden tai useamman ominaisuuden toimimattomuuden.

Bulma on CSS-kehys, jonka avulla sivuston tyylien tekeminen on huomattavasti nopeampaa, sillä se tarjoaa paljon valmiita luokkia, joita voi käyttää hyväksi sivuston ulkoasua luodessa.

CSS eli Cascading Style Sheet on merkintäkieli, jolla toteutetaan web-sivujen tyylittely.

Docker on ohjelma, joka virtualisoi käyttöjärjestelmätasolla niin sanottuja kontteja, joihin tarvittavat sovellukset ja niiden riippuvuudet asennetaan. Yksi palvelin voi pyörittää useaa tällaista konttia ilman huolta siitä, että ohjelmat voisivat jotenkin muiden samalla palvelimelle asennettujen ohjelmien toimintaa.

Eclipse on suosituin Java-kehitysympäristö. Se on ilmainen ja löytyy Windowsille, Linuxille ja Applen macOS-käyttöjärjestelmälle. Vaikka Eclipse on pääsääntöisesti Javalle tarkoitettu kehitysympäristö, löytyy siitä tuki usealle muulle ohjelmointikielelle.

Frontend on sovelluksen käyttöliittymä.

GIT on versionhallintatyökalu, jonka Linus Torvalds julkaisi vuonna 2005. Torvalds kehitti työkalun, koska hänen silloin käyttämänsä versionhallintatyökalu BitKeeper muuttui maksulliseksi. GIT:in etuina on muun muassa nopeus, yksinkertainen suunnittelu ja kyky hallita laajoja projekteja tehokkaasti (Git 2018).

HTML eli Hypertext Markup Language on merkintäkieli, joka on jokaisen web-sivun peruselementti.

Http-pyyntö / -kutsu on selaimesta lähtevä kutsu palvelimelle. Sekä http-kutsussa että sille tulevassa vastauksessa voidaan esimerkiksi lähettää tekstiä, kuvia tai ääntä.

IntelliJ IDEA on JetBrainsin kehittämä tekstieditori ohjelmistokehitykseen Javalla. IntelliJ:stä löytyy sekä ilmainen, että maksullinen versio. Vaikka IntelliJ on ensisijaisesti Java-kehitysympäristö, voi sitä käyttää myös muilla ohjelmointikielillä.

Java on Oraclen vuonna 1995 julkaisema ohjelmointikieli, joka on pysynyt suuressa suosiossa vuodesta toiseen, koska se on hyvin monikäyttöinen tarjoamansa Java Virtual Machinen (JVM) ansioista, mikä toimii lähes millä alustalla tahansa. Lisäksi Java on todella nopea kieli, mikä sekin on JVM:n ansiota.

JavaScript on jokaisen selaimen tukema ohjelmointikieli ja sen avulla toteutetaan websivujen toiminnallisuus. JavaScriptiä voi käyttää myös serveripuolen ohjelmakoodin toteuttamiseen käyttämällä Node.js:ää.

Maven on yksi Javan monista paketointityökaluista, jonka tarkoitus on helpottaa kehittäjien työtä pitämällä huolta projektin riippuvuuksista lataamalla ne automaattisesti (ja kaikki ne, joita lisätyt riippuvuudet tarvitsevat) kunhan ne lisätään projektin POM-tiedostoon.

MVP eli minimum viable product tarkoittaa tuotetta, missä ominaisuuksia on vain ja ainoastaan sen verran, että se riittää täyttämään ensimmäisten asiakkaiden vaatimukset.

MySQL on relaatiotietokanta, mihin yhden tai useamman sovelluksen tieto tallennetaan käyttöä varten. Tietokanta muodostuu tauluista ja niiden relaatioista keskenään.

Spring Boot on yksi osa Spring ohjelmistokehystä Javalle, jonka tarkoituksena on nopeuttaa web-kehitystä Javalla huomattavasti. Spring Bootin suurimmaksi eduksi voidaan laskea sen tekemä automaattinen konfigurointi monen asiassa, jotka joutuisi muuten konfiguroimaan itse.

SQL eli Structured Query Language on relaatiotietokantojen hallintaan tarkoitettu kieli.

Thymeleaf on työkalu, jolla voidaan rakentaa esimerkiksi HTML-sapluunoja (template) internetsivuja varten. Thymeleafia voidaan käyttää muun muassa silloin, kun tehdään backend-kehitystä Javalla eikä projektiin haluta ottaa mukaan JavaScript ohjelmistokehystä, vaan käyttöliittymäkoodi hoidetaan Thymeleafilla ja Javalla.

TypeScript on Microsoftin JavaScriptin päälle rakentama ohjelmistokieli, joka lisää kieleen valinnaisen vahvan tyypityksen, mikä JavaScriptistä puuttuu. Suosituimmista Ja-

vaScript ohjelmistokehyksistä vain Angular käyttää oletuksena TypeScriptiä, mutta sen voi ottaa käyttöön muissakin ohjelmistokehyksissä.

VSCode on Microsoftin kehittämä tekstieditori, joka on tarkoitettu sovelluskehitykseen. VSCode tukee useita ohjelmointikieliä ja sille löytyy todella paljon lisäosia, jotka helpottavat sovelluskehittäjän työtä.

2 Oman nykyisen työn analyysi

Työtehtäväni koostuvat vaatimusmäärittelyjen perusteella luotujen tehtävien ratkaisemisesta. Käytännössä tämä tarkoittaa ohjelmiston eri osien suunnittelemista, ohjelmointia ja testausta sekä integroimista muuhun koodiin. Tehtävät voivat olla mitä tahansa tietokantatöistä käyttöliittymän toteutukseen. Sen lisäksi osallistun ohjelmiston kokonaisarkkitehtuurin suunnittelemiseen ja asiakaspalavereihin, joissa käydään läpi nykytilannetta ja tulevaisuutta sekä mahdollisia ohjelmistoon kohdistuvia uusia vaatimuksia. Tämän hetkisessä projektissa olen myös mukana suunnittelemassa Excel-tiedostojen muotoilua ja nimeämistä, johon ohjelmalogiikka perustuu.

Työaikani on melko vapaasti valittavissa, joten voin tehdä työt yleensä omien aikataulujeni mukaan. Lisäksi minulla on mahdollisuus tehdä työtä muuallakin kuin asiakkaan tiloissa, mutta asiakkaan toivomus on, että työskentelen mahdollisimman paljon paikan päällä, jotta kommunikointi on nopeaa ja vaivatonta.

Työssäni tarvitaan laaja-alaista osaamista web-kehityksestä ja vaatimuksena onkin, että kykenen työskentelemään projektin kaikissa sovelluskehitykseen liittyvissä tehtävissä. Lähtötilanne tässä projektissa on ollut se, että olen rakentanut yksin sekä käyttöliittymää että palvelinpuolen koodia. Iso osa projektin alkuajasta onkin mennyt Excel-tiedostopohjan suunnitteluun, ohjelmalogiikan hahmottelemiseen ja palvelinpuolen ohjelmointiin. Työssäni minun pitää kyetä suoriutumaan tehtävistäni annetuilla työkaluilla, eli yleisimpien käyttöjärjestelmien (Windows, Linux, macOS) osaaminen on välttämätöntä. Lisäksi täytyy myös osata yleisimpiä ohjelmistokehitystyökaluja, kuten esimerkiksi Eclipse ja/tai Visual Studio Code, mitkä ovat monipuolisia tekstieditoreja ohjelmakoodin kirjoittamiseen.

Työhöni tarvittavan tietotaidon olen hankkinut sekä koulusta että itseopiskelulla. Työni kannalta välttämättömät sosiaaliset taidot olen hankkinut edellisistä työpaikoistani, joissa työtehtäväni olivat suurilta osin asiakaspalvelua. Ongelmaratkaisutaitoni ovat mielestäni hyvät, mikä onkin erittäin tärkeää, sillä työtehtäväni ovat asiakkaan esittämän ongelman ratkaisemisia tavalla tai toisella ja yleensä tiukan aikataulun sisällä. Tiukka aikataulu vaatii sen, että myös paineensietokyvyn täytyy olla hyvä. Osaamistani pyrin kehittämään koko ajan lisää opiskelemalla tarvittavia asioita vapaa-ajalla ja koodaamalla mahdollisimman paljon myös työpäivien ulkopuolella. Työaikaa saan käyttää kaksi tuntia viikossa ohjelmistokehitykseen liittyvien asioiden opiskeluun. Sen lisäksi töitä tehdessä osaamiseni kehittyy kuin itsestään, kun saan ratkaista uusia ongelmia joka päivä.

Työpaikkani ja -tehtäväni osaamisvaatimuksiin nähden osaamiseni on aloittelevan toimijan ja taitavan suoriutujan välimaastossa; tarvitsen työtehtävistäni suoriutumiseen kollegoiden apua, mutta kykenen myös toimimaan työtehtävissäni itsenäisesti pitkiäkin aikoja, vaikka usein työtovereideni apu voisi nopeuttaa prosessia huomattavasti. Nykyisessä projektissani tarvittavaa teknistä osaamista ovat muun muassa Javan ohjelmistokehys Spring Boot sekä JavaScriptin ohjelmistokehys Angular 4. Nämä tekniikat ovat minulle uusia, joten niiden sujuva käyttö vaatii vielä opettelua.

Vahvuuksinani ovat mielestäni hyvät ongelmanratkaisutaidot, kommunikointitaidot sekä kyky etsiä itse ratkaisua ongelmiin hyödyntämällä eri tietolähteitä. Tietolähteinä käytän suosittuja ja luotettavia sivustoja, kuten StackOverflow, Baeldung, Medium ja MDN web docs. Suurimmat vajavuudet osaamisessani löytyvätkin hyvien suunnittelumallien (design patterns) ymmärryksestä sekä kokonaisarkkitehtuurin suunnittelusta ja hahmottamisesta. Esimerkiksi ohjelman kokonaisarkkitehtuurin suunnittelu on siksi haastavaa, että en osaa hahmottaa, mitä ohjelma oikeasti tulee tarvitsemaan vaatimusmäärittelyn perusteella. Tämä johtaa helposti vääränlaisiin ratkaisuihin arkkitehtuuria suunnitellessa. Sen lisäksi vaatimusmäärittelyn läpikäyminen voi olla osaltani vajavaista, sillä en välttämättä aina hahmota, mikä on oikeasti mahdollista toteuttaa asiakkaan antaman aikataulun puitteissa.

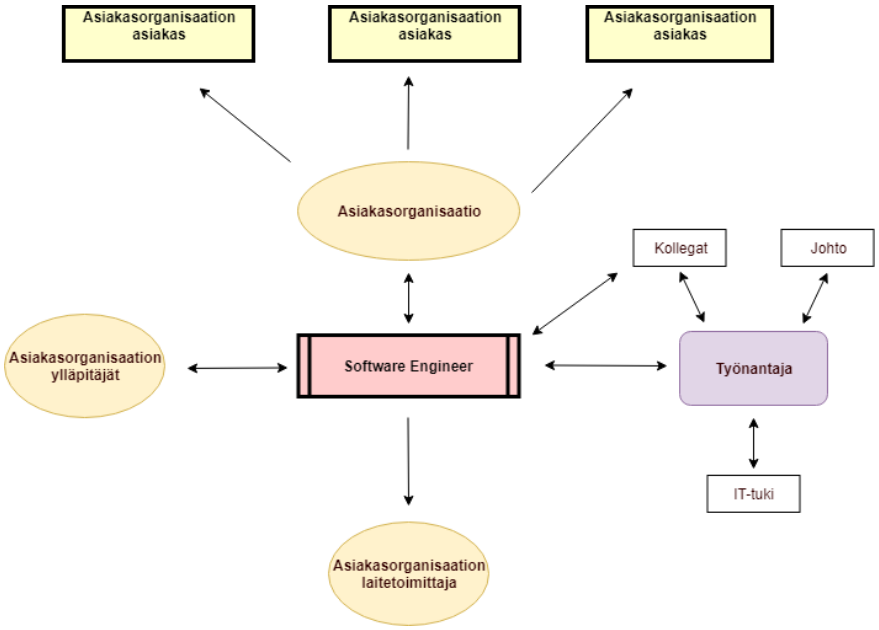
Olen vielä urani alussa, joten kehittymiselle on aikaa ja tarvetta. Ohjelmointikokemusta minulla on vuoden verran, mutta työkokemusta ohjelmoinnista tätä kirjoittaessa on vasta yhden kuukauden verran. Tämä näkyy päivittäin toiminnassani siten, että en ole yhtä tuottelias kuin kollegani ja osaamiseni ei ole samalla tasolla, vaan joudun hyödyntämään eri tietolähteitä huomattavasti useammin kuin työkaverini. Osaamistani kehitän kuitenkin jatkuvasti paremmaksi itseopiskelulla työajan ulkopuolella, mutta myös työtehtäväni ovat sellaiset, että osaamiseni kehittyy kuin itsestään. Työssäni pääsen päivittäin painimaan uuden ongelman parissa, joka yleensä vaatii jonkin pienen kokonaisuuden opiskelua, mikä taas kehittää yleistä osaamistasoani.

Olen ilmaissut haluni työnantajalle oppia Javaa ja sen ohjelmistokehyksiä, Spring MVC ja Spring Boot, ja päästä sellaisiin projekteihin mukaan, missä näitä tekniikoita on käytössä. Erityisen tärkeää minun olisi kehittyä itse ohjelmoinnissa mahdollisimman hyvin – ei niinkään erinäisissä tekniikoissa tai ohjelmointikielissä. Olen kuitenkin luottavainen siitä, että ymmärrykseni ohjelmoinnista ja siihen liittyvistä isoista kokonaisuuksista kehittyvät huimasti melko lyhyelläkin aikavälillä.

Minun pitäisi myös oppia ohjelmistojen virtualisoinnista (Docker) sekä järjestelmäautomaatiosta (Ansible), vaikka ne eivät vielä olekaan kovin suuressa osassa työtehtäviäni. Niiden käyttäminen on kuitenkin lähes pakollista nykypäivänä.

2.1 Sidosryhmät työpaikalla

Tärkein ulkoinen sidosryhmä työssäni on aina se asiakasorganisaatio (kuvio 1), jonka projektissa työskentelen. Työnantajani liiketoimintamalli on kehittää ja toimittaa web-sovelluksia asiakkaille, joten työni on miltei täysin eri asiakkaiden kanssa toimimista ja yhteistyötä, jotta asiakas saa juuri sellaisen web-sovelluksen kuin haluaa. Tärkeä osa jokaista työpäivää on kommunikoida asiakasorganisaation projektipäällikön kanssa projektin etenemisestä, mahdollisista ongelmista ja ohjelmiston kokonaisarkkitehtuurista ja kuinka se integroidaan asiakkaan jo mahdollisesti olemassa oleviin järjestelmiin. Asiakasorganisaatio myy mahdollisesti kehittämääni tuotetta eteenpäin omille asiakkailleen. Monesti projekteissa asiakasorganisaatio myös toimittaa sovelluskehitysvälineet, kuten tietokoneet, monitorit ja oheislaitteet, joilla se haluaa työn toteutettavan ja tarjoaa IT-tuen laitteilleen.



Kuvio 1. Sidosryhmät työssä

Tärkein sisäinen sidosryhmä on projektissa mukana olevat kollegani sekä esimieheni. Vaikka olen pääsääntöisesti tekemässä töitä asiakasorganisaation tiloissa, olen muidenkin kollegoideni kanssa tekemisissä eri kommunikointikanavien kautta sekä yhteisissä tapahtumissa toimistolla. Yrityksen johto on vastuussa uusien asiakkaiden hankkimisesta ja päättää kuka työskentelee missäkin projektissa ja kenen kanssa. Työntekijöillä on myös vaikutusvaltaa siihen, missä projektissa haluaa olla mukana eikä ketään pakoteta sellai-

seen projektiin, mikä ei itseä kiinnosta. Sisäinen IT-tuki on vastuussa laitehankinnoista ja niiden tuesta.

2.2 Vuorovaikutustaidot työpaikalla

Olen päivittäin tekemisissä vähintäänkin niiden työkavereiden kanssa, jotka ovat samassa projektissa minun kanssani. Keskustelemmekin projektin tehtävistä, vastuista ja etenemisestä jatkuvasti ja autamme toisiamme aina tarpeen tullen. Yleisen kommunikoinnin lisäksi myös arvioimme toisten työn jälkeä ja annamme parannusehdotuksia, jos sellaisia tulee mieleen. Meillä ei ole varsinaista prosessia siihen, että miten koodi päätyy tuotantoon, vaan jokainen laittaa tekemänsä muutokset versionhallintaan. Puhumme ja suunnittelemme myös ohjelmiston kokonaisarkkitehtuuria ja eri tekniikoiden sopimisesta kyseiseen projektiin. Käymämme keskustelut eivät aina liity millään tavalla työhön tai työn tekemiseen vaan juttelemme usein myös ihan muista asioista.

Päivittäin jaamme myös projektiin liittyviä työtehtäviä keskenämme, jotta välttyisimme päällekkäiseltä työltä. Kommunikointi tässä asiassa onkin erittäin tärkeää, sillä on hyvin turhauttavaa huomata tehneensä turhaa työtä useita tunteja. Työtehtävien jakaminen on yleensä hyvin vapaamuotoinen prosessi eikä meillä ei ole käytössä tiketöintijärjestelmää vaan meneillään olevat ja tulevat tehtävät kirjataan korkeintaan tussitaululle. Monesti työtehtävien jako onkin lyhyt keskustelu ja kesken päivän joku voi ilmoittaa vapaasti ottavansa uuden tehtävän työn alle.

Asiakkaan kanssa kommunikointi on helppoa ja joustavaa, koska nykyisessä projektissani asiakkaan projektinvetäjä istuu samassa huoneessa kanssani. Keskustelu onkin yleensä hyvin vapaamuotoista. Tämän lisäksi on projektiin liittyen yleensä vähintään yksi palaveri viikossa, missä on muitakin asiakkaan henkilöitä mukana ja nämä tapaamiset ovatkin hieman virallisempia. Näissä palavereissa on yleensä tarkoituksena selvittää eteen tulleita ongelmia niiden henkilöiden kanssa, joilla on valtaa tehdä ratkaisuja sen suhteen, että miten edetään. Kun tekeillä oleva ohjelmisto alkaa olla valmis, pidämme useita palavereita asiakkaan ylläpitotiimin kanssa, missä on tarkoitus siirtää tieto kehittäjiltä ylläpidolle. Pidän yhteyttä päivittäin pikaviestityökalu Slackin avulla myös sellaisiin työkavereihini, joiden kanssa en työskentele samassa tilassa päivittäin. Keskustelu Slackin eri kanavilla onkin hyvin vilkasta ja juttuseuraa tai apua löytyy aina. Lisäksi työnantajani järjestää monia tapahtumia työntekijöilleen, jossa pääsee tapaamaan myös muualla työskenteleviä kollegoita. Yleensä nämä tapahtumat ajoittuvat perjantaihin ja niistä hyvinä esimerkkeinä on työnantajan kustantama joka perjantainen aamupala ja sauna, jonne kaikki ovat tervetulleita.

3 Päiväkirjaraportointi

3.1 Seurantaviikko 15

Maanantai 9.4.2018

Päivän tavoitteena oli edistää käyttöliittymän toiminnallisuutta sekä parannella ulkoasua. Aloitin päivän tarkistamalla kalenterin mahdollisten kokousten varalta ja lukemalla edellisen viikon koodia muistaakseni mihin olin jäänyt edeltävän viikon perjantaina. Sen jälkeen tarkistin sähköpostit asiakkaan postien varalta ja ryhdyin töihin.

Käyttöliittymän kehitystyön aloitin rakentamalla etusivulla olevan hakutoiminnon toimintalogiikan sekä tyylittelemällä sen. Tarkoituksena oli tehdä hakutoiminto niin, että kun käyttäjä alkaa kirjoittamaan tekstikenttään, alapuolelle ilmestyy vaihtoehtoja, mihin käyttäjän syöte sopisi. Näistä näkyvistä tuloksista ensimmäinen olisi valittuna oletuksena.

Päivän toinen tehtävä olikin Excel-taulukoiden mallin suunnittelua asiakkaan ja kollegoideni kanssa. Kokous kesti yhteensä noin tunnin verran ja päädyimme lopulta aika suuriin muutoksiin.

Iltapäivä menikin käyttöliittymän yhden osan uudelleen suunnittelussa ja toteutuksessa. Päädyin myös rakentamaan kaksi pientä, mutta uudelleenkäytettävää komponenttia: tekstikenttä ja valikko. Kumpaakin komponenttia voidaan jatkossa käyttää missä tahansa osassa käyttöliittymää syöttämällä niille haluttua dataa.

Asettamani päivätavoite täyttyi useampaankin kertaan käyttöliittymää uudelleen rakentaessa. Päivän aikana opin rakentamaan pieniä, mutta modulaarisia komponentteja, joita voidaan hyödyntää jatkossa suhteellisen helposti.

Tiistai 10.4.2018

Asetin päivän tavoitteeksi saattaa valmiiksi hakukomponentin toiminnallisuuden ja tyylittelyn. Päivä oli kuitenkin hyvä aloittaa tarkistamalla sekä sähköposti että kalenteri, jotta voisin suunnitella päivän ohjelman tarkemmin. Tänään oli tiedossa vain yksi sisäinen infotilaisuus ja sekin vasta iltapäivällä, joten pystyin aloittamaan päivän tehtävät.

Vaikka tavoitteena olikin tehdä hakukomponentti täysin valmiiksi, päädyin aloittamaan eilen tehtyjen lapsikomponenttien parantelulla. Kumpaankin komponenttiin piti lisätä ta-

pahtumien lähetys vanhemmalle, jotta käyttäjän antamaa tietoa voitaisiin käyttää. Tämä osoittautuikin helpoksi tehtäväksi; riitti kun vähän luin Angularin omaa dokumentaatiota, jossa opastettiin käyttämään ”@Output” – notaatiota ja EventEmitter –luokkaa, jolla voidaan lähettää data vanhemmalle (Google 2018a).

Saatuani lapsikomponentit lähes valmiiksi siirryin päivän varsinaisen tavoitteen kimppuun. Hakukomponentin parissa vierähtikin suurin osa päivästä enkä edes saanut sitä valmiiksi. Haun toimintalogiikan implementointi ei ollutkaan niin yksinkertainen tehtävä kuin olin aluksi virheellisesti kuvitellut. Haun tekstikentän alapuolella piti näyttää suodatettu lista tuloksista käyttäjän syötteen perusteella ja tuloksia piti voida klikata, jotta se valitaan. Lisäksi, jos käyttäjä painaisi enteriä tai tabulaattoria, pitäisi listasta valita ensimmäinen tulos. Suurin haaste olikin päästä käsiksi suodatetun listan dataan, mikä oli tekstikentän ulottumattomissa alkuperäisellä toteutustavallani. Aikani aherrettuani haun parissa sain sen toiminnallisuuden välttävälle tasolle, mutta edessä on vielä paljon työtä sen kanssa.

Hakukomponenttia parannellessani keskustelin myös asiakkaan kanssa ohjelman liiketoimintasääntöjen pohjana toimivan Excelin muodosta ja vastailin kysymyksiin parhaani mukaan.

Päivän lopussa osallistuin työnantajani infotilaisuuteen ja valmistauduin henkisesti, että siinä menisi sille allokoitu tunti kokonaan. Ilokseni infosessio loppuikin jo alle puolessa tunnissa, joten pääsin vielä oikeiden töiden pariin. Lopetin päivän hahmottelemalla edistyspalkkikomponenttia ja sen toteutusta ja toimintalogiikkaa.

Keskiviikko 11.4.2018

Aloitin päivän siirtymällä suoraan koodin kimppuun ja jätin sähköpostit omaan arvoonsa. Olin edellisenä päivänä jo visioinut mielessäni uuden komponentin ulkoasua ja toimintaa ja paloin halusta päästä tekemään sitä. Edistyspalkkikomponentin parissa vierähti huomaamatta pari tuntia ja sainkin sen hyvälle alulle.

Sähköposteissa ei ollut mitään tähdellistä, mikä olisi vaikuttanut päivän ohjelmaan, joten jatkoin edistyspalkin kehittämistä. Puolen päivän aikoihin pidimme palaverin kollegoideni ja asiakkaan kanssa Exceleistä ja loppujen lopuksi päädyimme aika suuriin muutoksiin, mutta ne muutokset eivät onneksi vaikuta käyttöliittymän toimintaan. Olisi ollut ikävää, jos muutokset olisivat vaatineet sen, että osia käyttöliittymästä tai sen toiminnallisuudesta olisi pitänyt tehdä uudelleen.

Päivän tavoitteenani oli saada edistyspalkki valmiiksi, mutta en ihan onnistunut saavuttamaan sitä, sillä iltapäivällä keskityin parantelemaan hakukomponentin toimintaa sekä viimeistelemään tyylittelyä tulos- ja yhteenvetokomponenttiin.

Torstai 12.4.2018

Aloitin päivän samalla tavalla kuin edellisenkin eli aloin koodaamaan. Asetin mielessäni tavoitteeksi saada päivän aikana hakutulosten listauksen toimimaan kunnolla ja edistää lopputulosten näyttämiseen tarkoitettua komponenttia. Hakutuloslistassa ongelmana oli se, että hiirtä liikuttaessa listan päällä myös ”selected” -teksti liikkui mukana, vaikka valinta ei oikeasti muuttunut. Se, että teksti liikkui hiiren mukana, johtui aikaisemmin tekemistäni valinnoista, jotka pitäisi korjata.

Loppujen lopuksi en joutunut muuttamaan tekemääni koodia hirveästi, että sain listan toimimaan kuten pitääkin. Lisäsin myös samalla niin sanotun ”placeholderin” hakutulosten eteen, minkä tilalle on tarkoitus laittaa hakutulosta vastaava kuva kunhan vain saamme kuvat käyttöömme.

Päivällä siirryinkin tekemään tuloskomponenttia. Hetken pähkäiltyäni päätin hajottaa sen kahteen osaan eli vanhempi- ja lapsikomponentteihin ja muutin samalla myös ulkoasua. Olin aikaisemmin laittanut hakutuloksen omaan laatikkoonsa, mutta muutin laatikon kortiksi (card), jotta lopputulos olisi selkeämpi ulkoasultaan. Lapsikomponentin toteutin samaan tapaan kuin aikaisemmatkin eli se on tarkoitettu vain sille annetun datan näyttämiseen.

Huomasin iltapäivällä kalenteria katsoessani, että perjantaina sovellusta pitäisi esitellä asiakkaalle, joten päätin laittaa tuloskomponentin linkit toimintaan. Ehdin saada linkeistä valmiiksi vain yhden, jota klikkaamalla päästään tekemään uusi haku ja vanhat tiedot häviävät. Aloitin myös tekemään toisen linkin toiminnallisuutta, mistä päästään takaisin hakuun, mutta aikaisemmin käytetyt hakukriteerit säilyisivät ja ne olisivat valmiina tekstikentissä ja valintalaatikoissa.

Perjantai 13.4.2018

Jatkoin heti aamusta käyttöliittymän kehitystä, koska tiesin, että iltapäivällä sovellusta pitäisi esitellä asiakkaalle. Päätin jatkaa töitä datan säilyvyyden kanssa. Tai säilyvyys ei ollut ongelma, mutta sen välittäminen valintalaatikoihin osoittautui yllättävän suureksi haasteeksi. Ahersin ongelman kanssa yli tunnin verran, mutta sain sen ratkaistua lopulta,

jonka jälkeen käytin sovelluksen testaamiseen noin vartin verran aikaa, jotta näin, että kaikki toimii varmasti niin kuin pitääkin.

Kun olin saanut sovelluksen testauksen mielestäni tarpeeksi kattavaksi, lisäsin yhteenvetoon muutaman puuttuvan kohdan ja parantelin ulkoasua hieman ja vastaamaan enemmän suunnittelijan piirtämiä kuvia. Samalla lisäsin testidataa muutenkin esittelyä varten, jotta sovelluksen toiminnasta saisi paremman kuvan.

Palaverissa, jossa sovellusta esiteltiin, menikin melkein kaksi tuntia ja siellä tuli esille, että lopputulosten näyttämiseen tarkoitetun sivun ulkoasu muuttuisi aika paljon. Esittelytilaisuudessa keskusteltiin pitkään väliotsikoista ja datan sijoittelusta sekä siitä tehtävän PDF-tiedoston ulkoasusta.

Sovellusesittelyn jälkeen aloin tekemään sovittuja visuaalisia muutoksia lopputuloskomponenttiin. Pääsin hyvään alkuun eikä seuraavalle viikolle jäänyt paljoa tekemistä sen suhteen, vaikka lopetinkin päivän noin tuntia aikaisemmin normaaliin verrattuna.

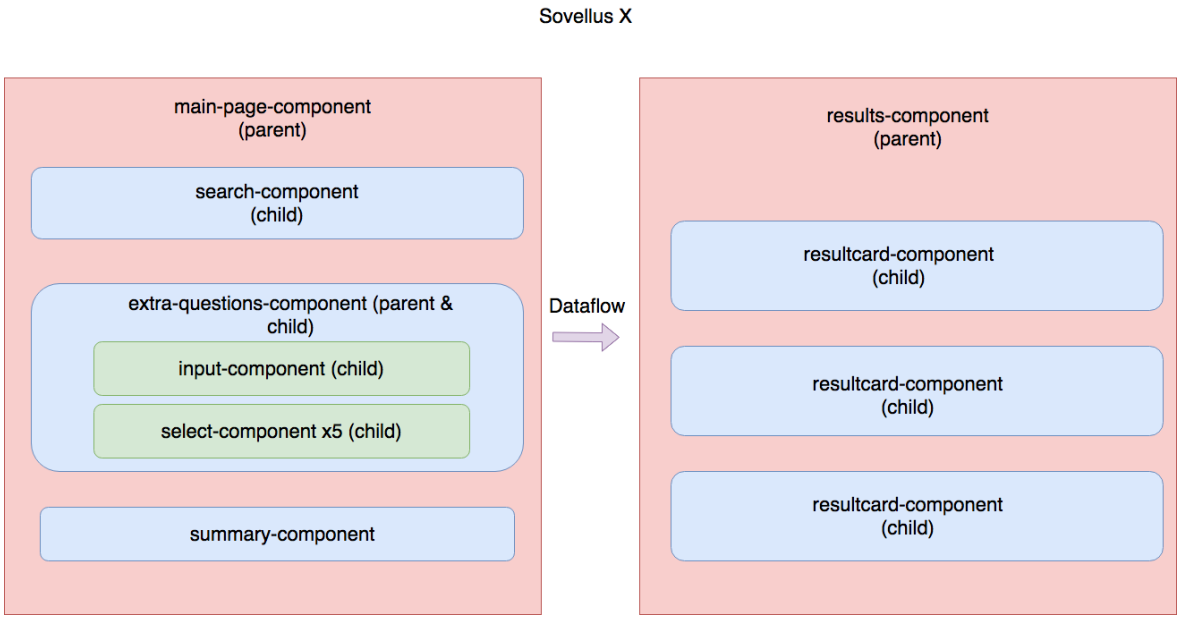
Viikkoanalyysi

Kuluneen viikon aikana sain tehdä paljon minulle ennalta tuntemattomia asioita ja jouduin opettelemaan paljon uutta. Viikon aikana en koskenut kertaakaan backendin koodiin – ainoastaan välillä katselin selän takana kollegani touhuja. Tämä asia tulee todennäköisesti muuttumaan jatkossa ja varsinkin kun ulkoasun suunnittelija saa sovelluksen kuvat valmiiksi ja pääsee myös tekemään käyttöliittymän ohjelmakoodia.

Viikon suurimpana ongelmana oli datan välitys lapsikomponenteille, sen käsittely ja lopulta hyödyntäminen käyttäjän joko palatessa takaisin sivulle tai tekemällä uuden haun aikaisemmilla valituilla hakukriteereillä.

Aiheutin itselleni lisää työtä aikaisemmilla valinnoillani tekemällä monen komponentin osalta toiminnallisuuden yhteen komponenttiin, vaikka parempi ratkaisu olisi ollut jakaa toiminnallisuus useampiin pieniin komponentteihin jo heti alusta asti. Onneksi sain kuitenkin komponentit jaettua jälkikäteenkin, mutta se tietysti tarkoitti lisää työtä ja ennen kaikkea sellaista työtä, miltä olisi voitu välttyä. Tässä kohtaa kokemattomuuteni ohjelmoinnista näkyy, sillä uskoisin kokeneen sovelluskehittäjän tehneen heti kättelyssä oikeat ratkaisut toiminnallisuuden jakamisesta ja käyttöliittymän kokonaisarkkitehtuurista (kuvio 2). Ei olisi mitenkään mahdotonta, että käyttöliittymän toiminta ja arkkitehtuuri tulisi muuttumaan,

kunhan vain kollegani alkaa tekemään koodia siihen. Hänellä on kuitenkin paljon enemmän kokemusta siitä asiasta.



Kuvio 2. Sovelluksen käyttöliittymän tämän hetkinen yksinkertaistettu arkkitehtuurikuvaus.

Huomasin myös viikon aikana sen, että kun ohjelmoi työkseen (ja varsinkin muiden henkilöiden kanssa) niin mikään ei ole niin pysyvää kuin muutos. Omissa sivuprojekteissani olen tottunut siihen, että kun jotain päätöksiä teen, en niitä muuta kovin helposti. Mutta tällaisessa oikeassa projektissa, missä on useita asiakkaan edustajia ja kollegoita mukana (sekä asiakkaan henkilökuntaa että kollegoita), muutoksia tulee todella usein ja joskus ne voivat olla yllättävän suuriakin. Näin myös sen, että vaikka asiasta on jo tehty päätös ja sen mukaan edetty, ei se tarkoita sitä, että se olisi lopullinen päätös. Tämä lienee sitä ketterää ohjelmistokehitystä. Vaikka ketterät menetelmät tekevät kehityksestä joustavaa, on siinäkin ongelmansa. Bloombergin mukaan ketteriä menetelmiä käytettäessä ei keskitytä ohjelman arkkitehtuuriin tai uudelleen käytettävän koodin kirjoittamiseen (Bloomberg 2015). Sen vähän kokemuksen perusteella, mikä minulla on, voin hyvin yhtyä tähän mielipiteeseen. Projektissa ei asiakkaan puolelta kukaan tunnu keskittyvän ohjelman arkkitehtuuriin vaan se on jätetty kokonaan meidän vastuulle ja koska aikataulu on tiukka, ei ole aina aikaa suunnitella kirjoitettavaa ohjelmakoodia tarpeeksi, että se olisi mahdollisimman hyvin uudelleenkäytettävissä. Vaikka teoriassa ketterät ohjelmistokehitysmenetelmät kuulostavat hyvältä, en itse ole vielä ehtinyt nähdä mitään mainittavia etuja. Kuvittelisin kuitenkin, että tämä tulee muuttumaan, kunhan saan enemmän työ- ja projektikokemusta.

Tämä jatkuva muutos onkin ehkä yksi haasteellisimmista osista ohjelmointikehittäjän työtä. Toinen haasteellinen asia on mahdolliset kommunikaatiovaikeudet asiakkaan ja kehit-

täjien välillä. Jos kumpikaan ei ymmärrä toista, on lopputuloksena todennäköisesti vain hukattua aikaa ja rahaa, kun tuote ei vastaakaan esitettyjä vaatimuksia.

Seurasin myös sivusta, kun kokeneempi kollegani ratkaisi asiakkaan esittämän ongelman. Ratkaisu poikkesi omasta ratkaisustani täysin, enkä olisi kyseistä ratkaisua osannut itse ajatellakaan. Luonnollisesti hänen esittämä ratkaisunsa oli elegantimpi ja pitkällä tähtäimellä huomattavasti helpompi ylläpitää ja jatkokehittää – joko meidän tai toisen kehitystiimin toimesta. Ohjelmistokehitystä tehdessä ikinä ei saisi unohtaa pyrkiä siihen, että sovelluksen jatkokehitys tai ylläpito olisi mahdollisimman helppoa. Karu totuus on kuitenkin se, että vaikka itse olisi se henkilö, joka tulee ylläpitämään sovellusta, niin kirjoitetun koodin ymmärrys haalistuu ajan kuluessa.

Vaikka luinkin paljon esimerkiksi Angularin omaa dokumentaatiota, jouduin valitettavan usein turvautumaan Googlen ja muiden ihmisten apuun ongelmien ratkaisussa. Esimerkkinä olkoon vaikka se kun luin lapsi-vanhempi -komponenttien välisestä suhteesta ja miten tietoa välitetään komponentilta toiselle, jouduin etsimään malliesimerkkejä vielä dokumentaation tueksi, jotta onnistuin soveltamaan sitä omaan koodiini. Syynä tähän ei ollut dokumentaation vajavaisuus tai se, että dokumentaatio olisi ollut kirjoitettu epäselvästi, vaan lähinnä oman ymmärryksen puutteellisuus. Angularin dokumentaatio on kuitenkin omasta mielestäni todella laadukasta ja pääsääntöisesti riittävän kattavaa. Tämä on sellainen asia, missä haluan kehittyä ja missä minun pitää ehdottomasti kehittyä paremmaksi. Uskonkin, että tulen siinä asiassa kehittymään kokemuksen karttuessa ja kykenen paremmin soveltamaan virallisen dokumentaation mahdollisia ohjeita.

Tietysti ei sovi unohtaa, että dokumentaation lukemisessa kehittyy samaan tapaan kuin ohjelmoinnissa eli mitä enemmän sitä tekee, sitä paremmin sen ymmärtää. Koska olen varma, että en ole ainoa tämän ongelman kanssa painiva ihminen, yritin etsiä internetistä vinkkejä kuinka kehittyä tässä asiassa. Aika nopeasti vastaan tulikin mielenkiintoinen blogikirjoitus, missä annettiin kahdeksan vinkkiä dokumentaation lukemiseen aloittelijoille. Näistä kahdeksasta kolme tärkeintä on mielestäni kärsivällisyys, useat lähteet ja tutustu termeihin (Barard 2017). Kärsivällisyys on ominaisuus, mikä on erittäin tärkeä ohjelmistokehityksessä ylipäätään, sillä monesti vastaantulevat ongelmat ovat sellaisia, että niihin ei löydy ratkaisua ihan hetkessä. Useiden lähteiden käyttö on monesti välttämätöntä, koska dokumentaatio ei aina esimerkiksi sisällä tarpeeksi informaatiota. Termeihin tutustuminen tuli uutena ja hyvänä vinkkinä minulle, sillä usein käy niin, että en tiedä, mitä jokin termi tarkoittaa ja sitä saatetaan käyttää todella paljon, joten moni asia voi jäädä ymmärtämättä.

Kaiken kaikkiaan viikko meni mielestäni hyvin ja saavutin päiväkohtaiset tavoitteet useasti. Olin tuottelias koko viikon ja sain edistettyä käyttöliittymää enemmän kuin alkuviikosta kuvittelinkaan. Yhteistyö kollegoideni ja asiakkaan kanssa toimi hyvin eikä sen suhteen ongelmia esiintynytkään koko viikon aikana. Onnekseni projektissa mukana ovat osaavat työkaverit, jotka ovat valmiita auttamaan minua silloin kun apua tarvitsen. Viikko meni siinäkin mielessä hyvin, että sain opetella paljon uusia asioita ja nähdä erilaisia tapoja ratkaista sama ongelma. Osaamiseni ja ymmärrykseni Angularin kanssa paranee päivä päivältä ja toivottavasti niin tulee käymään myös Javan ja Spring Bootin kanssa, kunhan vain pääsen tekemään niitäkin.

3.2 Seurantaviikko 16

Maanantai 16.4.2018

En asettanut päivälle mitään tavoitteita aamulla. Kalenterin mukaan edessä oli sprintin suunnittelukokous, joten halusin katsoa ensin sen läpi ennen kuin alkaisin miettimään, että mitä kannattaisi edistää. Käytin aamulla tunnin verran perjantaina tehdyn työn tarkistamiseen ja jatkoin hieman tulossivua. Suunnittelukokouksen jälkeen asetin tavoitteeksi yhteenvetosivun parantelun.

Iltapäivällä edistin tulossivua hieman lisää, mutta käytin suurimman osan ajasta kollegoideni tekemän edellisen projektin käyttöliittymän koodin tutkimiseen. Tarkoituksena olisi implementoida autentikointi ja autorisaatio samaan tapaan kuin siinä sovelluksessa, joten halusin perehtyä lähdekoodiin tarkasti. Autentikoinnilla tarkoitetaan keinoa varmistaa, että kuka käyttäjä on ja onko käyttäjä varmasti se kuka väittää olevansa. Autorisaatiolla puolestaan tarkoitetaan sitä, että varmistetaan onko käyttäjällä oikeuksia resurssiin tai sivuun. Koodia jonkin aikaa luettuani, ymmärrykseni parani siitä prosessista, mutta joudun varmasti vielä keskustelemaan kyseisen henkilön kanssa, joka koodin teki, että ymmärsin varmasti oikein. Lisäksi pitää myös selvittää, että voinko edes aloittaa käyttöliittymäpuolella autentikointia, autorisaatiota ja sivujen suojaamista ennen kuin serveripuolella on ehditty tekemään asian eteen jotain. Valitettavasti serveripuolen koodi on viivästynyt pahasti, sillä kollega, jonka sitä pitäisi pääsääntöisesti tehdä ja edistää, on joutunut käyttämään suurimman osan ajasta edellisen projektin defektien korjaamiseen ja selvittelyyn.

Voisin itsekin ehkä edistää backendiä hieman, mutta työ on tällä hetkellä sellaisessa vaiheessa, että koen paremmaksi jättää sen osaavammalle kollegalle, jotta arkkitehtuuri menee kerralla oikein. Töitä lopetellessa huomasin, että en ollut muistanut parannella yhteenvetosivua, joten päivän tavoite jäi saavuttamatta.

Tiistai 17.4.2018

Aamulla töihin kävellessä asetin mielessäni päivän tavoitteeksi lisäkysymyskomponentissa olevan tekstikentän alle tulevan hakutuloslistan toiminnallisuuden tekemisen. Töihin päästyäni aloitin kuitenkin etusivun hakukentän alla olevan niin kutsunut ”top results” – listan viimeistelyllä. Listassa tulee näkymään (tällä hetkellä kovakoodatuilla arvoilla) 14 suosituinta hakukriteeriä. Olin jo aikaisemmin tehnyt listan niin, että se näkyy, mutta mitään toiminnallisuutta sillä ei vielä ollut. Parin tunnin sisällä sain toteutettua mielestäni hyvän tavan näyttää listan ja miten se reagoi käyttäjän painalluksiin. Samalla vaivalla myös muokkasin etusivun toimintaa niin, että jos halutaan lisätä kolmas hakukriteeri, kaksi oletuksena näkyvää hakukenttää menee piiloon ja sen sijaan näytetään jo niihin syötetyt arvot.

Sen jälkeen siirryinkin päivän varsinaisen tavoitteen kimppuun. Koitos osoittautuikin hieman hankalammaksi kuin kuvittelin, sillä tekstikenttä oli lapsikomponentin tuottama ja sen alle tuleva listaus tuloksista oli vanhemman hallussa. Päädyinkin lopulta siirtämään myös listauksen lapsikomponentin vastuulle. Hommaan meni kokonaisuudessaan ehkä tunnin verran.

Testatessa sovelluksen uusia toimintoja huomasin bugin, joka täytyisi korjata välittömästi. Käytin kokonaisuudessaan melkein kolme tuntia kyseisen bugin selvittämiseen, mutta sain onneksi sen ratkaistua omin voimin. Virhe oli sellainen, että kun hakukentän alla olevan valintaruudun klikkasi niin, että se ei ole valittuna, kaikki annetut hakukriteerit hävisivät. Valinnan poistamisella valintaruudusta oli tarkoituksena nollata vain yksi kriteeri.

Iltapäivällä ehdin vielä tehdä etusivun HTML:n lähes täysin uusiksi hyödyntämällä edellä mainittua lapsikomponenttiin rakennettua hakua. Lopputuloksena etusivun HTML:stä tuli huomattavan paljon tiiviimpi ja vähemmän itseään toistava, joten olin hyvin tyytyväinen lopputulokseen.

Keskiviikko 18.4.2018

Keskiviikon tavoitteeksi asetin saada hakutuloksiin näkyviin kyseiseen tulokseen liittyvän kuvan. Olin aikaisemmin laittanut vain paikan kuvalle ja kollegani oli maanantaina tehnyt pieniä muutoksia, jotta kuva tulisi näkyviin, mutta se ei toiminut.

Tutkimme työkaverini kanssa yhdessä ongelmaa ja lopulta löysimmekin syyn, joka oli se, että paketti, mistä kuvat ovat, käytti väärää polkua kuvien hakemiseen. Kun syy löytyi, oli korjauskin helppo.

Kun kuvat oli saatu toimimaan, huomasin pari virhettä toiminnassa ja ulkoasussa. Ensinnäkin hakutulos meni kahdelle riville, jos se oli liian pitkä ja se näytti huonolta. Toisekseen enterin tai tabin painaminen hakukenttään kirjoittamisen jälkeen ei toiminut. Syy toimimattomuuteen oli helppo löytää ja se johtui siitä, että olin unohtanut implementoida niiden toiminnallisuuden kokonaan.

Huomasin, että työkaverini oli lisännyt yhden uuden elementin hakukenttään ja sille kutsuttavan metodin. Tai lähinnä vain metodin nimen, sillä itse toiminnallisuus uupui, joten päätin toteuttaa sen. Kyseessä oli nappi, jota painamalla hakukenttä tyhjentyy. Tämän kanssa ilmeni yllättäviä ongelmia, sillä kun hakukentän tyhjensi, se samalla aktivoi metodin, mikä oli tarkoitettu tabin painamiseen. Pienen selvittelyn jälkeen sain tämän ratkaistua.

Iltapäivä menikin asiakkaan työntekijän läksijäisissä, joten päivän kokonaistuottavuus ei noussut kovin korkeaksi.

Torstai 19.4.2018

Tämän päivän tavoitteeksi asetin korjata mahdollisimman paljon niin sanottuja tyylivirheitä sivustolta eli niitä kohtia, jotka eivät vastanneet suunnittelijan tekemiä kuvia. Olin aamupäivän etänä ja koin olevani hieman tuotteliaampi kuin mitä asiakkaan tiloissa paikan päällä ollessa. Aloitin päivän työt lisäämällä tarkentavien kysymysten sivulle yhden uuden kysymyksen, mikä jo kuvista löytyi, mutta uupui käyttöliittymästä. Samalla laitoin sen näkymään myös yhteenvetosivulle. Sen jälkeen järjestelin kumpaakin sivua uudelleen niin, että ne vastasivat kuvia.

Kaiken kaikkiaan koko aamu meni pieniä muutoksia ja korjauksia tehdessä, mutta sain monet kohdat vastaamaan suunnitelmia, joten olin tyytyväinen aamun ja aamupäivän suorituksiin.

Iltapäivän käytin suunnittelijan kanssa ulkoasua läpikäydessä ja tulevaisuuden suunnittelemisessa. Hän on vihdoin päässyt irtautumaan toisesta projektista kokonaan (ainakin molemmat toivomme niin) ja pystyy nyt keskittymään kunnolla tähän projektiin, joten alamme tehdä enemmän yhteistyötä keskenämme. Hän oli jo ehtinytkin tehdä muutoksia

ja korjauksia ulkoasuun. Vielä ennen työpäivän lopettamista, aloittelin reitityksen suunnittelemista ja toteuttamista.

Päivän tavoite tuli kyllä saavutettua moninkertaisesti, sillä sain tehtyä paljon parannuksia sivuston ulkoasuun.

Perjantai 20.4.2018

Tavoite perjantaille oli saada valmiiksi automaattinen täyttö hakukenttään, mitä työkaverini oli edellisenä iltana jo aloittanut. Ideana oli, että kun käyttäjä alkaa kirjoittamaan hakukenttään jotain, automaattinen täyttö tarjoaa tekstille sitä loppua, mikä on hakutuloksissa ensimmäisenä. Luonnollisestikaan se ei saanut tarjota mitään, jos ensimmäisen tuloksen ensimmäinen kirjain ei ollut sama kuin käyttämän laittama kirjain. Ominaisuus oli paljon haastavampi toteuttaa kuin mitä osasin ajatellakaan.

Tekstikentissä ei ole ominaisuutena automaattista täyttöä ja tekstin piti olla hieman erivärinen kuin käyttäjän kirjoitus eikä sanan alkuosa saa tulla näkyviin – vain sen kohdan pitäisi näkyä, mikä lopettaa käyttäjän syötteen. Ratkaisu tähän oli asettaa hakukentän alle uusi kenttä, mihin automaattitäytön teksti tulee ja siitä tekstistä piti leikata yhtä monta kirjainta alusta pois kuin mitä käyttäjä oli jo ehtinyt kirjoittamaan. Lisää haastetta toi myös se, että automaattitäytön teksti piti olla täsmälleen oikeassa kohdassa, että se näytti aidolta.

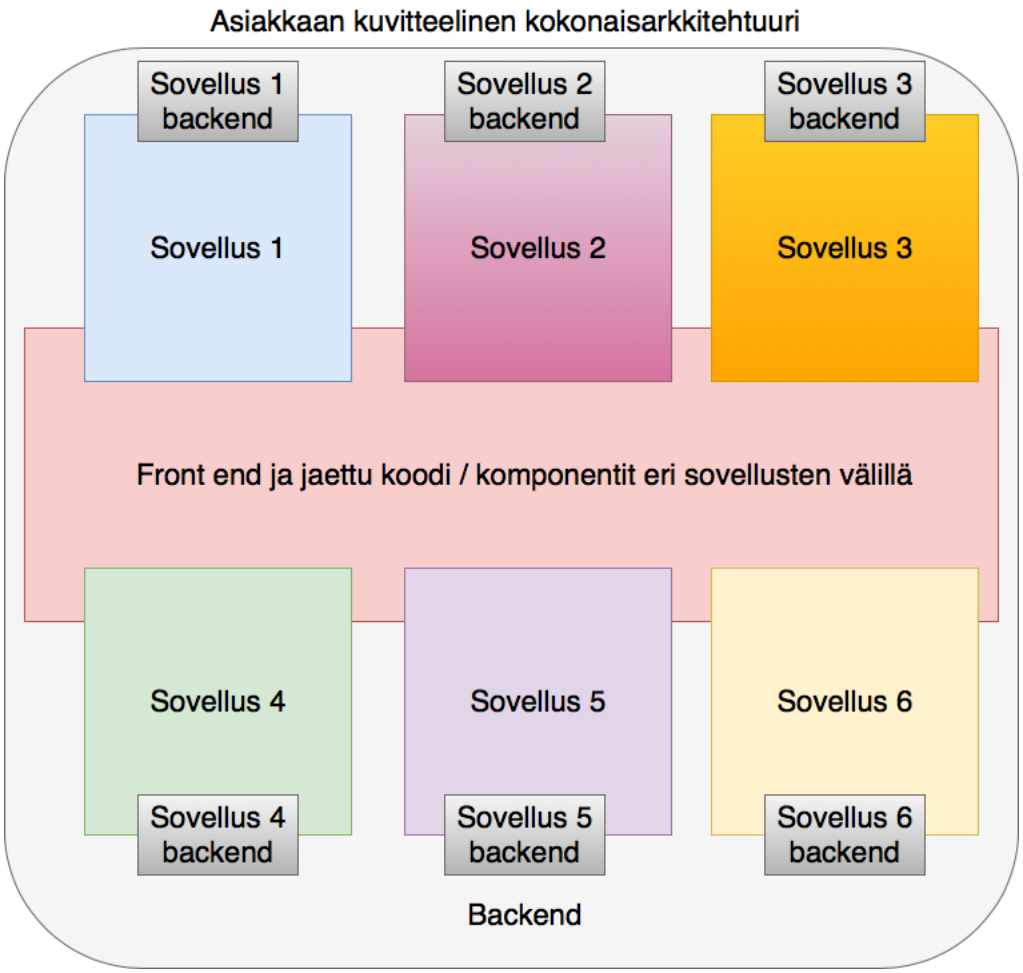
Iltapäivällä oli pienimuotoinen palaveri asiakkaan edustajan kanssa heidän arkkitehtuurista ja siinä tehdyistä ratkaisuista. Tällä hetkellä ympäristö on tehty niin, että heidän taustajärjestelmään upotetaan uusia sovelluksia, mitä esimerkiksi teemme, mutta jo ensimmäisen upotetun sovelluksen kohdalla oli jo ilmennyt suuria ongelmia. Aina, kun sovellukseen lisättäisiin uusia toiminnallisuuksia, pitäisi testata sekä sovellus että taustajärjestelmä, joten tuotantoonvienti on todella hidas prosessi. Lisäksi tuotantoon vienti tulee hidastumaan jokaisen uuden sovelluksen kohdalla ja lopulta menee ihan mahdottomaksi, kun upotettuja sovelluksia alkaa olla useampia. Koska varsinaisia toteutettavia ratkaisuehdotuksia ei tullut esille, täytyy pohdintaa jatkaa vielä.

Vielä ennen kotiinlähtöä tuli pyyntö tehdä myyntiä varten kuva lopputulossivusta tietyillä hakukriteereillä ja tuloksilla, jotta sitä voitaisiin esitellä potentiaaliselle asiakkaalle.

Viikkoanalyysi

Tällä viikolla pääteemana oli sovelluksen kokonaisulkoasun korjaaminen ja saattaminen näyttämään kuvia vastaavaksi. Sen lisäksi korjailin paljon bugeja ja siistin koodia mahdollisimman paljon, jotta turhalta toistolta vältyttäisiin.

Viikon suurin onnistumisen tunne tuli ehdottomasti hakukentän automaattisentäytön onnistuneesta toteutuksesta. Olin sen tekemistä pohtinut jo pidemmän aikaa, että miten sen saisi parhaiten toteutettua ja mielestäni onnistuin siinä hyvin.



Kuvio 3. Asiakkaan kuvitteellinen kokonaisarkkitehtuuri.

Eniten keskustelua tämän viikon aikana käytiin asiakkaan kokonaisarkkitehtuurista (kuvio 3) ja siihen tehdyistä ratkaisuista ja niiden järkevyydestä. Asiakkaan tekemä valinta on ymmärrettävä siinä mielessä, että sen toteutus alkuvaiheessa on paljon helpompi ja asiat saadaan nopeammin tuotantoon, mutta pidemmän päälle se ei ole kestävä arkkitehtuurillinen ratkaisu. Ensi viikolla ainakin yksi työkavereistani kokoustaa yhden työnantajani kokeneen sovellusarkkitehdin kanssa siitä, että miten asiakkaan haluamat asiat olisivat järkevintä toteuttaa. Varmaa on se, että päädyttiin mihin ratkaisuun tahansa, eivät muutokset tule tapahtumaan kovin nopealla aikataululla, vaan puhutaan useamman viikon kestäväs-

tä kehitysprosessista. On hyvin todennäköistä, että asiakas ei osta meiltä kehittäjiä tekemään sitä vaan luottaa omiin sovelluskehittäjiinsä, mikä voi venyttää prosessin kestoa entisestään johtuen siitä, että he ovat jo nyt todella kiireisiä.

Olen päässyt seuraamaan aitiopaikalta mihin väärät arkkitehtuurilliset ratkaisut voivat johtaa. Muutokset arkkitehtuuriin voivat tulla hyvin kalliiksi eikä nyt edes puhuta mistään suurista virheistä. Oppi kyllä meni ainakin itselle perille, vaikka vain sivustaseuraaja olenkin. Kokonaisarkkitehtuurin kunnollinen suunnittelu on kultaakin kalliimpaa siinä vaiheessa, kun puhutaan vähänkin laajemmasta kokonaisuudesta.

Positiivista tässä viikossa oli myös se, että kollegani pääsi tekemään jo osan viikosta kanssani tätä sovellusta, joten projekti eteni paljon enemmän kuin mitä se olisi edennyt, jos olisin ollut ainoa kehittäjä tällä viikolla. Harmillisesti se henkilö, joka olisi pääasiallisessa vastuussa backendin koodista, ei ole ehtinyt käytännössä ollenkaan tekemään mitään tähän projektiin liittyvää, vaan hän on joutunut jatkuvasti korjaamaan edellisen projektin tuotannon kriittisiä bugeja. Alamme kohta olla siinä tilanteessa, että projekti ei pääse etenemään ollenkaan ennen kuin serveripuolen koodia saadaan tehtyä. Tulin asiakkaalle niin kutsuttuna full stack kehittäjänä eli että tekisin töitä sekä käyttöliittymässä että palvelinpuolella. Olen palvelinpuolelle jotain pientä jo tehnytkin asiakkaalla ollessani, mutta ratkaisut ovat olleet lähinnä MVP tyyppisiä eivätkä ne tulisi olemaan kovin kestäviä tuotannossa.

Asiakkaan arkkitehtuuri frontendin (käyttöliittymä) suhteen on sinänsä mielestäni toteutettu oikein ja Angularin dokumentaation mukaisesti. Ongelmana onkin ehkä se, että sovelluksilla on jaettua koodia, mikä ei ole suositusten mukaista vaan koodit olisivat suotavaa pitää erillään, jotta yllä kuvatun kaltaisilta ongelmilta vältytään (Google 2018a). Hyvä puoli oli kuitenkin itselle se, että en ollut tällaista toteutusta aikaisemmin nähnyt enkä ollut sellaista edes osannut ajatellakaan, mikä on laajentanut omaa näkemystäni eri sovellusarkkitehtuureista. Odotan sekä innolla, että kauhulla minkälaisia erilaisia sovellusarkkitehtuurikokonaisuuksia tulen vielä työurallani näkemään. Tämä tuskin on edes huonoimmasta päästä ja tämän kanssa tuntuu olevan paljon ongelmia.

3.3 Seurantaviikko 17

Maanantai 23.4.2018

Päivän tavoitteena oli edistää sivuston ulkoasun siistimistä ja muokkaamista enemmän suunnitelmakuvien suuntaan. Aloitin päivän parantelemalla tulossivua sekä generoimalla

asiakkaalle yhden kuvan tietyillä hakutuloksilla myyntiä varten. Viime viikon lopussa ilmeni outoja ongelmia, jotka näyttivät jatkuvan vielä tänäänkin. Käyttämämme CSS-kehyksen, Bulman, tyylisäännöt joko ohittivat meidän tekemät muokkaukset tai ylikirjoittivat ne kokonaan. Ongelmaan ei tuntunut auttavan mikään. Monet CSS-säännöt eivät vain tulleet voimaan, vaikka teimme mitä tahansa kollegani kanssa.

Iltapäivän käytin kokonaan sopivan kirjaston etsimiseen serveripuolelle. Asiakkaan yksi vaatimus on, että tulossivusta saadaan tehtyä PDF-tiedosto, joka voidaan lähettää halutuun sähköpostiosoitteeseen. Ongelmana oli löytää sopiva kirjasto PDF-tiedoston luomiseen. Kaikki vaihtoehdot olivat joko todella vanhoja, maksullisia tai vaativat koko lähdekoodin julkaisemisen tai sitten liian uusia, jotta niitä voitaisiin ottaa käyttöön. Pitkän selvittelyn jälkeen näyttää kuitenkin siltä, että joudumme turvautumaan melko vanhaan kirjastoon, koska parempaa ei vain tunnu olevan tarjolla.

Tiistai 24.4.2018

Tiistain tavoitteeksi asetin itselleni saada PDF-tiedoston luomiseen tarvittavat kirjastot valittua sekä keinon testata sen toimivuus. Koska vaihtoehdot olivat melko vähissä kirjaston valinnan suhteen, päädyin ainakin näin alkuun ottamaan Flying Saucer -nimisen kirjaston ja sieltä väliaikaisesti sen artefaktin eli JAR-paketin, joka käyttää iText 2.x versiota PDF:n luomiseen. Valitettavasti iText 2 on niin vanha, että sen tilalle on varmaankin pakko etsiä toinen kirjasto tulevaisuudessa. Nyt on kuitenkin oleellisempaa saada backendin osalta MVP-tasoinen ratkaisu valmiiksi tulevia demoja ajatellen. Hyvää on se, että kirjasto tukee myös CSS:ää, joten luotu PDF-tiedosto saadaan näyttämään todennäköisesti lähes identtiseltä varsinaisen tulossivun kanssa, mikä on hyvä projektin yhteneväisen ulkonäön kannalta.

Itse kirjaston käyttöönotto hoitui lisäämällä projektin riippuvuuksiin kirjaston nimi ja versio. Positiivista oli se, että löysin toimivan esimerkin kirjaston käyttämiselle, joten testitapauksen rakentaminen oli helppoa. Pieniä muutoksia esimerkkikoodiin ja sain rakennettua toimivan ratkaisun.

Lisäksi erittäin hyvä asia oli se, että tämä ratkaisu tukee Thymeleaf HTML-sapluunojen käyttöä, joten sinne saa upotettua suoraan Java-muuttujia ja PDF-tiedosto voidaan koota useammasta eri HTML-sapluunasta. Muokkailin sapluunaa lopuksi kevyesti vähän siihen suuntaan kuin miltä sen pitäisikin näyttää, mutta sapluunan lopullinen muoto jäänee ulkoasun suunnittelijan vastuulle.

Keskiviikko 26.4.2018

Laitoin itselleni päivän tavoitteeksi siivota aikaisemmin tekemääni serveripuolen koodia. Monet tekemäni asiat olivat muuttuneet tarpeettomiksi kollegani aloitettua tekemään toisenlaista menettelytapaa Excel-tiedostojen lukemiseen ja käsittelyyn. Poistin kaikki turhat metodit ja muutin Java-luokkia selkeämmiksi.

Kun olin omasta mielestäni saanut siistittyä kaiken tarpeettoman koodin pois, siirryin pohtimaan sähköpostin lähettämistä. Sovelluksesta piti olla mahdollista tehdä hakutuloksista PDF-tiedosto, joka sitten lähetetään sähköpostilla tekijälle. Onneksi Javalle löytyy todella paljon erilaisia kirjastoja, koska se on ohjelmointikielenä jo melko vanha, joten ratkaisu löytyi suhteellisen helposti. Löysin vielä kaiken lisäksi aihion, minkä päälle on hyvä lähteä rakentamaan toiminnallisuutta.

Loppupäivästä kävimme vielä suunnittelijan kanssa keskustelua sivuston ulkoasusta ja mihin suuntaan lähdemme sitä viemään.

Päivä ei ollut kovin tuottava, mutta saavutin kuitenkin tavoitteeni ja sain siivottua palvelinpuolen koodista turhat asiat pois, jotta jatkokehitys olisi helpompaa ja selkeämpää.

Torstai 26.4.2018

Päivän tavoitteenani oli edistää sähköpostin lähetystä sekä tutustua käyttöliittymään kollegani tekemiin koodimuutoksiin, mutta tietotekniikka heitti kapuloita rattaisiin. En koko päivänä saanut kehitysympäristöä toimimaan, joten en päässyt kirjoittamaan riviäkään koodia tai tutkimaan käyttöliittymän muutoksia käytännössä. Koko päivä meni siihen, että yritin kaikin keinoin saada ympäristön toimimaan tuloksetta.

Perjantai 27.4.2018

Perjantain tavoite oli kaikessa yksinkertaisuudessaan saada kehitysympäristö toimintaan, jotta voisin jatkaa oikeiden työtehtävieni tekemistä. Sain kuin sainkin ympäristön toimimaan, mutta vasta kun päivästä oli jo mennyt puolet.

Iltapäivällä oli pieni demo asiakkaalle, missä esiteltiin sovelluksen nykytilannetta. Esityksen aikana huomasin muutamia bugeja, joten käytin loppupäivän niiden ratkomiseen. Päivän saavutukset jäivät aika laihoiksi, mutta ainakin pääsin jatkamaan töitäni.

Viikkoanalyysi

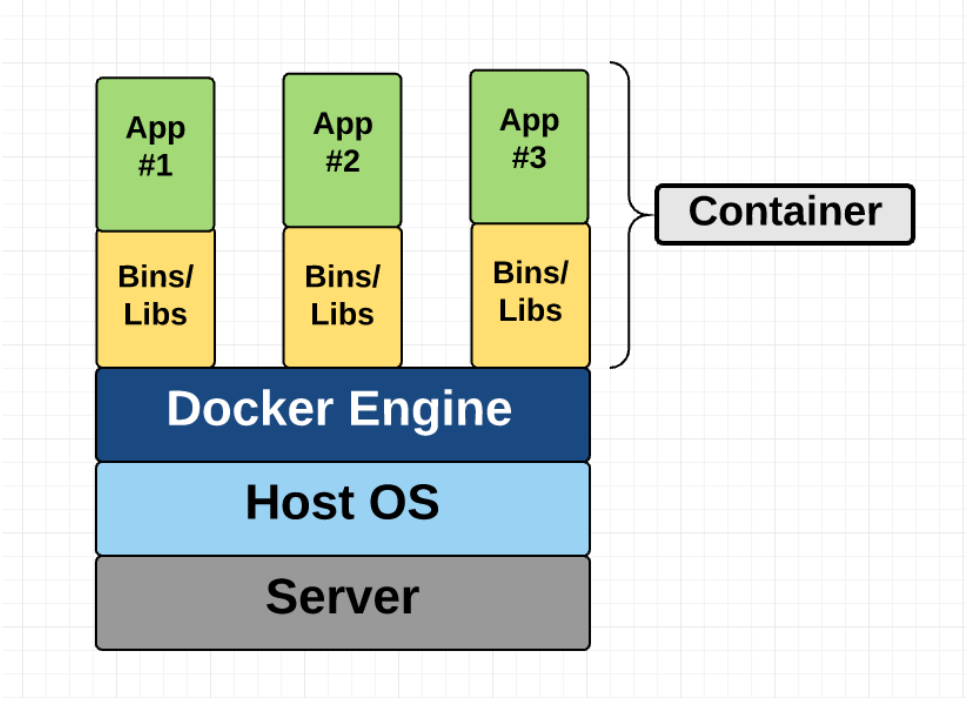
Viikko erosi siinä mielessä kahdesta edellisestä, että tällä viikolla edistin serveripuolen koodia hieman ja pääsin tekemään asioita Javalla TypeScriptin sijaan ja käytin todella paljon aikaa ongelmaselvitykseen, joka ei ollut millään tavalla tekemisissä itse tuotekehityksen kanssa. Kokemukseni Java-projekteista on kuitenkin hyvin kevyellä tasolla. Oikeastaan voisi sanoa, että ennen tätä projektia olen tehnyt Javalla vain sen, mitä koulussa on opiskeltu eli ohjelmoinnin perusteet ja ohjelmistoprojektikurssin työ, mikä oli websovellus tietokantaa hyödyntäen. Mistään Java ohjelmistokehyksestä minulla ei ollut kokemusta, mikä oli harmillista. Yritin opiskella hieman Javaa ja ennen kaikkea Spring Boot ohjelmistokehystä ennen projektin alkua, mutta en ehtinyt tehdä sitä kovin paljoa.

Kun viikkoa pysähtyy ajattelemaan, nousee ensimmäisenä mieleen ajatus ja tunne, että tämä viikko ei ollut kovin tuottoisa enkä saanut edistettyä asiakkaan projektia juuri ollenkaan. Eli viikosta on jäänyt melko negatiivinen muistijälki. Tällainen on kuitenkin väistämätöntä, sillä ei jokainen viikko voi olla pelkkää hyvin sujunutta kehitystyötä. Joskus voi mennä pitkäänkin yhden ominaisuuden tekemisessä tai ongelman ratkaisemisessa ja viikon saldo tuotetun koodirivimäärän suhteen voi olla todella pieni. Se, montako riviä ohjelmakoodia, on muutenkin todella huono mittari tuottavuuden suhteen. Onneksi sellaista ei vielä ole tullut vastaan, että esimerkiksi asiakas seuraisi projektin edistymistä kirjoitettujen rivien määrällä – muuten kuin internetin ihmeellisessä maailmassa. Välillä tuntuu, että se on pelkkä urbaani legenda.

Käytin viikolla myös paljon aikaa dokumentaation lukemiseen, joista esimerkkeinä voisi mainita kollegani valitsema kirjasto nimeltä n-cube, jota voisi käyttää muun muassa Excel-tiedostojen datan tallentamiseen helppokäyttöiseen muotoon (DeRegnaucourt 2018) sekä Dockerin virallinen dokumentaatio. Valitettavasti n-cuben dokumentaatio ei ole kovin laaja eli käytännössä yksi readme -tiedosto GitHubissa ja ainakin tällainen vasta-alkaja, kuten minä, kaipaisi vähän syvällisempää dokumentaatiota ja mielellään esimerkkejä toiminnasta.

Docker (kuvio 4) puolestaan tarjoaa hyvän ja laajan dokumentaation monien esimerkkien kera. Tätä tulikin luettua paljon, kun yritin selvittää loppuviikon kehitysympäristön toiminnallisuuteen liittyviä ongelmia. Sivuilta löytää erittäin hyvin tietoa lähes kaikkeen Dockeriin liittyvästä ja monelle eri käyttöjärjestelmälle. Lisäksi dokumentaatioon on sisällytetty erillinen aloitusosuus, jonka avulla jokainen pääsee hyvään alkuun Dockerin käytössä (Docker 2018a).

Jos viikosta haluaa jotain positiivista sanoa, niin ainakin tuli hiottua taitoa lukea dokumentaatiota hieman paremmaksi, mitä minun onkin pitänyt kehittää. Ja jos haluaa edelleen jatkaa positiivista ajattelua, niin tämä työhän on pohjimmiltaan puhdasta ongelmanratkaisua, joten siinä mielessä loppuviikko ei mennyt ihan hukkaan, kun yritin saada kehitysympäristöä toimimaan. Tietysti olisin käyttänyt mieluummin aikaani kehitystyöhön ja siihen liittyvään ongelmanratkaisuun. Toivottavasti ensi viikolla selviän ongelmitta ja voin keskittyä työhöni.



Kuvio 4. Dockerin toimintaperiaate kuvana (Kasireddy 2016)

Docker on ehdottomasti yksi sellaisista teknologiasta, mitä ilman ei selviä web-sovelluskehityksessä nykypäivänä. Dockerin ehkä suurin hyöty on se, että sillä päästään eroon surullisen kuuluisasta minun koneellani se toimii -ongelmasta. Tämä korjaantuu sillä, että koko sovelluskehitys tapahtuu Dockerin luoman kontin sisällä ja tämän kontin voi kopioida sellaisenaan muille kehittäjille ja tuotantoon. Lisäksi kun sovellukset ovat omissa konteissaan ja kaikki niiden tarvitsemat riippuvuudet asentuvat kontin sisälle, eivät ne voi aiheuttaa ongelmia toisille sovelluksille (Docker 2018b).

Sovelluskehittäjien työn pitäisi ainakin teoriassa helpottua, kun jokaisella on sama ympäristö käytössä. Ei ole merkitystä, mitä koneelle on asennettu, kun sovellus ja sen tarvitsemat riippuvuudet ovat kaikki omassa erillisessä kontissaan. Itselläni on kuitenkin tästä hieman vaihtelevat kokemukset, sillä olen joutunut taistelemaan asiakkaan kehitysympäristön kanssa todella paljon, vaikka jokainen osa kehitysympäristöstä on omassa kontissaan.

Siitä huolimatta olen sitä mieltä, että Dockerista on enemmän hyötyä kuin haittaa – varsinkin kun puhutaan yritysmaailmasta. Yrityksellä voi olla todella monia eri sovelluksia samalla palvelimella ja silloin on melkein väistämätöntä, että sovellukset voivat aiheuttaa ongelmia toisilleen. Näenkin Dockerin sellaisena teknologiana, jonka haltuun ottaminen on minulle erittäin tärkeä asia urakehitykseni kannalta ja mitä varhaisemmassa vaiheessa sen onnistun tekemään, sitä parempi. Nykyisen kokemukseni perusteella en kuitenkaan ottaisi Dockeria käyttöön heti serveripuolen koodia tehtäessä Javalla, sillä sovelluksen uudelleenpaketointi ja Dockerin uudelleenkäynnistys on huomattavasti hitaampi prosessi, kuin Java-kehitysympäristöstä suoraan ajettuna. Mieluummin itse tekisin niin, että siirtäisin vasta valmiin sovelluksen Docker-konttiin ja korjaisin mahdolliset rikki menneet asiat.

3.4 Seurantaviikko 18

Maanantai 30.4.2018

Vapaa.

Tiistai 1.5.2018

Vappu

Keskiviikko 2.5.2018

Pitkän viikonlopun jälkeen on hyvä aloittaa työt kevyesti, joten tavoitteeksi asetin päivälle saada tehtyä lisäkysymyssivulla esiintyvän hakuehdotuslaatikon tehtyä niin, että sen esiintulo ei siirrä muita sivulla näkyviä valikoita ollenkaan, vaan tulee olemassa olevien elementtien päälle.
Aloitin kuitenkin päivän lukemalla sähköpostit sekä kollegoideni käymän keskustelun maanantailta. Siihen menikin yllättävän paljon aikaa, sillä maanantai oli ollut kiireitä täynnä, mutta onneksi mikään ei liittynyt tähän projektiin, missä itse olen mukana.

Seuraavat pari tuntia käytin laatikon muokkailuun sekä satunnaisiin bugikorjauksiin, mitä käyttöliittymästä löysin, kun testailin laatikkoa. Sen jälkeen poistin turhia muuttujia ja palveluita koodista, joita ei enää tarvittu, kun työkaverini oli tehnyt sivun navigoinnin uusiksi lopulliseen muotoonsa.

Iltapäivän käytin pakkaamiseen ja muuttamiseen, sillä saimme tietää, että neuvotteluhuoneen varaus, jossa työskentelimme, oli mennyt umpeen eikä sitä ollut uusittu, joten meidän piti siirtyä sieltä pois iltapäivän aikana. Valitettavasti vapaata tilaa ei ollut eikä tule olemaan ennen 10.5., joten joudumme kaikki olemaan etänä siihen asti. Kun pakkaus oli hoidettu, piti vielä metsästää asiakkaan IT-puolen henkilöitä, jotta saimme vielä kaikille toimivat VPN-yhteydet etätöitä varten.

Torstai 3.5.2018

Ensimmäisen etätyöpäivän tavoitteeksi asetin itselleni tehdä serveripuolelle metodin, joka ottaisi vastaan parametreina käyttöliittymästä lähetetyt hakutulokset, muodostaisi niistä PDF-tiedoston ja lopulta lähettäisi sen takaisin käyttöliittymään ladattavaksi.

Mutta jotta se ei olisi ollut liian helppoa, käytin ensin useamman tunnin pelkästään siihen, että saisin kehitysympäristön toimimaan. Pitkän taistelun jälkeen sain kuin sainkin kaiken toimimaan ja pääsin aloittamaan työt. Ehdin saada sen valmiiksi ja korjata käyttöliittymästä vielä etenemispalkin, joka oli mennyt rikki eilen, kun olin poistanut omasta mielestäni turhia osia koodista ennen kuin kehitysympäristö lakkasi toimimasta. Koska en jaksanut enää taistella sen kanssa, lopetin päivän hieman aikaisemmin kuin aion. Onneksi kuitenkin ehdin saavuttaa tavoitteeni.

Perjantai 4.5.2018

Päivän tavoite oli saada kehitysympäristö jälleen toimimaan. Aloitin ongelmanselvittelyn puoli kuuden aikaan aamulla ja vasta yhdeksän jälkeen sain hommat toimimaan. Olin jo aikaisemmin päättänyt, että teen tänään lyhemmän työpäivän eli tarkoituksenani oli lopettaa työt puolen päivän aikoihin, joten aikaa varsinaiselle kehitystyölle ei jäänyt kovin paljoa, ottaen huomioon, että taukojakin pitäisi pitää.
Kun vihdoin pääsin töihin, kollegani pyysi minua korjaamaan pari bugia, jotka olivat ilmestyneet jossain vaiheessa. Ensimmäinen oli, että tulossivulta uusi haku – nappia klikkaamalla eivät hakukriteerit enää nollautuneet ja toinen oli etenemispalkin toiminnallisuudessa. Hakukriteerien nollautumattomuus oli täysi mysteeri. Jostain syystä sille tehty metodi ei vain suostunut tyhjentämään enää annettuja kriteerejä. Pähkäilin sen kanssa yli tunnin ennen kuin tajusin, että se ei ole edes enää tarpeellinen ominaisuus. Tai siis ominaisuutena on toki tarpeellinen, mutta toimintalogiikka sen ympärillä oli käynyt turhaksi kollegani lisättyä hakuparametrit URL-osoitteeseen. Keksin, että voin ottaa hakukriteerit sieltä sen sijaan, että tallentaisin niitä applikaation omaan muistiin ja sitä kautta myös niiden nollaus onnistuisi helposti.

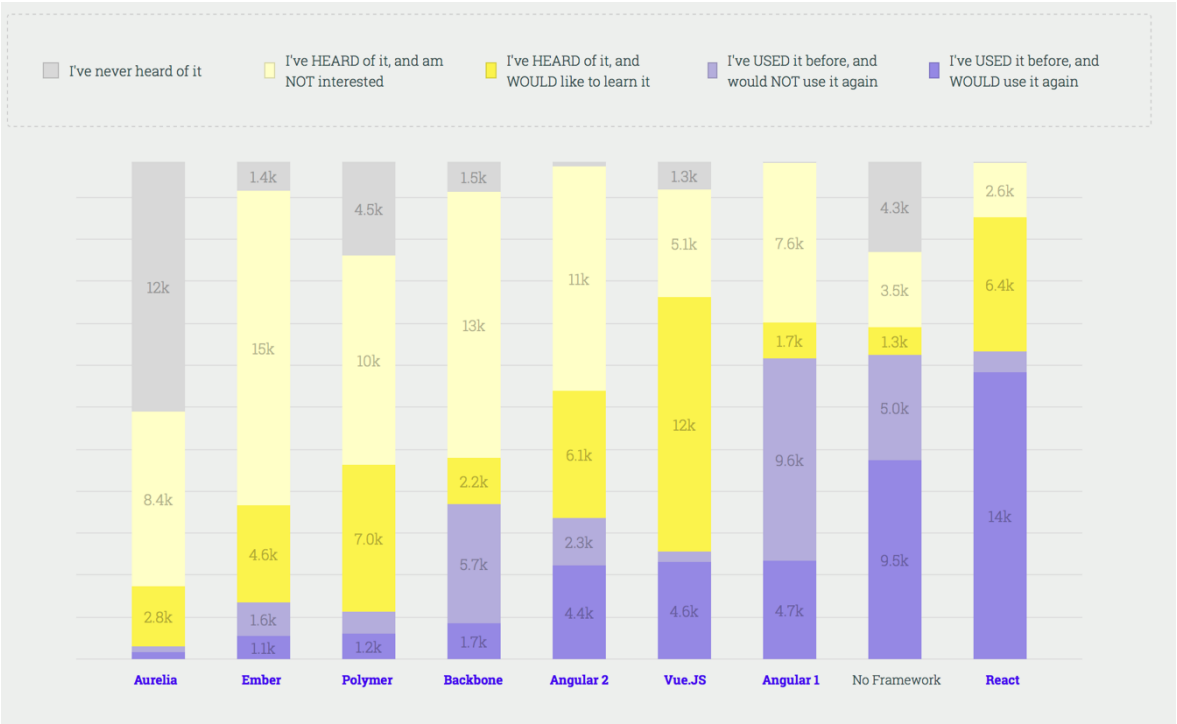
Käytin noin 30 minuuttia siihen, että poistelin turhia metodeja ja rakensin uutta toiminnallisuutta. Pienen testailun jälkeen totesin ratkaisun toimivaksi ja siirryin toisen bugin kimppuun. Tämä osoittautui helpoksi korjaukseksi, sillä olin unohtanut lisätä yhden ehdon sijainnin tarkistukselle, joten siihen ei mennyt kauaa aikaa.

Loppupäivän, jota oli noin 40 minuuttia jäljellä, käytin serveripuolen keskeneräisten testien poistoon, sillä ne estivät sovelluksen paketoinnin ja keskustelin työkaverini kanssa siitä, että miten toteuttaisimme hakujen seurannan, jotta niistä voitaisiin laskuttaa asiakkaan asiakkaita heidän toiveensa mukaisesti.

Viikkoanalyysi

Viikon saldo jäi melko kevyeksi johtuen vapaista ja ongelmista kehitysympäristön kanssa. Viikko koostui melkein pelkästään bugikorjauksista, mikä on tietysti ymmärrettävää kun projekti etenee hyvään tahtiin eikä käyttöliittymässä enää pitäisi olla suuria puutteita toiminnallisuuksien suhteen. Onneksi projektissa aloitti yksi uusi henkilö meiltä, joka tekee serveripuolta, mikä on jäänyt melko vähälle huomiolle. Käyttöliittymääkään ei saa edistettyä määräänsä enempää ennen kuin on oikeaa dataa saatavilla, sillä saatu data tulee muuttamaan asioita vähän.

Osaamiseni Angularissa on kyllä noussut kohisten tämän projektin aikana. Lähtötilanne oli kuitenkin se, että en ollut Angularilla tehnyt mitään muuta kuin kokeillut sitä hyvin kevyesti. Ennen kuin hyppäsin projektiin mukaan, ostin Udemystä yhden kurssin Angularista ja suoritin sen. Kurssin aikana käytiin perusasiat läpi ja tehtiin asiakkaiden hallintaan tarkoitettu pieni sovellus. Toki pohjalla oleva perusosaaminen toisesta JavaScript-ohjelmistokehyksestä, Vuesta, auttoi paljon, vaikka kokemus ei työelämästä ollutkaan. Karkeasti ottaen (JavaScript) ohjelmistokehyksistä voisi sanoa samaa kuin eri ohjelmointikielistä: kun osaat yhden, seuraavan opettelu on todella paljon helpompaa. Hyvä näin, koska JavaScript-ohjelmistokehyksiä on todella paljon, mutta onneksi vain kolme on laajemmassa käytössä. Kuvioista 5 nähdään, että näistä kolmesta suosituin on Facebookin kehittämä React, toisena Googlen Angular (Angular 1 + Angular 2) ja kolmantena Evan Youn kehittämä Vue (State of JavaScript 2017).



Kuvio 5. JavaScript ohjelmistokehysten suosio (State of JavaScript 2017)

Näistä kolmesta Vue on nuorin ja ainoa, jolla ei ole suurta yritystä takanaan, mikä voi olla esteenä sille, että suuremmissa yrityksissä sitä ei välttämättä haluta ottaa käyttöön. Vuen suuri suosio johtuu muun muassa sen helppokäyttöisyydestä, nopeudesta ja keveydestä (Kothari 2018). Oman kokemuksen perusteella voin allekirjoittaa ainakin helppokäyttöisyyden, sillä ymmärsin Vuen perusteet nopeasti ja vaivatta, kun aloin harjoittelemaan sen käyttöä. Toistaiseksi olen käyttänyt Vue ohjelmistokehystä vasta omissa projekteissani, mutta toivon pääseväni käyttämään sitä vielä työelämässä ja uskon, että se on vain ajan kysymys johtuen Vuen kasvavasta suosiosta.

Olen vasta ohjelmointiurani alussa ja varmaan myös osittain siitä syystä nautin työstäni todella paljon. Mutta totuus on, että tämä on ensimmäinen työelämän projekti, jossa olen mukana kehittäjänä ja mikä parasta, tämä oli täysin uusi projekti eikä vanhan sovelluksen ylläpitoa. Jossain vaiheessa kuitenkin on väistämätöntä, että jopa uuden projektinkin kanssa tähän työhön voi puutua ajan kanssa. Saati sitten kun pitää ylläpitää isoa sovellusta ja tekee vain bugikorjauksia. Siitä on luomisen tuoma nautinto kaukana. Puutumisen lisäksi voi myös uupua siihen, että on jatkuvasti opeteltava uusia asioita ja teknologioita, sillä tietotekniikka kehittyy niin nopeasti. Se, mikä on tänään uutta, on huomenna jo vanhaa teknologiaa.

Blogissa McEwen (2017) kirjoittaa, että ohjelmoiminen työkseen kahdeksan tuntia päivässä on täysin eri asia kuin sama harrastuksena. Projektit eivät välttämättä ole kovin mie-

lenkiintoisia tai niistä näkee vain pienen osan, koska kehitystyö on jaettu niin laajalle porukalle. Hänen mukaansa yritykset eivät hae mainostamaansa intohimoa koodaamiseen, vaan sillä tarkoitetaan sitä, että henkilö on valmis tekemään pitkiä päiviä ja sen lisäksi viikonloppuisin omalla ajallaan jatkaisi näiden ongelmien ratkaisua.

Jäin pohtimaan tätä kohtaa, että voisiko se olla totta. Olen seurannut vierestä, kuinka työkaverini on tehnyt todella pitkiä päivä (ja viikkoja) projektissa ja kuunnellut ehkä osittain totuudenmukaista nurinaa siitä, että ei jaksaisi tai että voisi vaihtaa työpaikkaa. Olen kuitenkin varma, että hän laskee leikkiä suurimmaksi osaksi, ainakin työpaikan vaihtamisen suhteen, mutta tuskin hänkään jaksaa jatkaa loputtomiin meidän näkökulmasta valmiin projektin bugikorjausten parissa, joiden olisi pitänyt jo siirtyä ylläpitotiimin vastuulle. Ja on hän jo minunkin aikana ehtinyt käyttää yhden viikonlopun asiakkaan arkkitehtuuriongelman ratkaisemiseen – ilman korvausta omalla ajallaan.

Voisiko tämä sattua omalle kohdalle? Varmasti. Olen intohimoinen työni suhteen ja ylipäätään pidän töiden tekemisestä, oli ammatti mikä tahansa. En näe sitä mitenkään utopistisena, että löytäisin itseni tekemästä ilmaista työtä viikonloppuisin vain siksi, että työpäivästä loppuvat tunnit kesken. Mutta tekstin lukeminen sai minut miettimään asioita. Ehkä on hyvä pitää selkeät rajat työn ja muun elämän välillä, jos ei muun takia niin edes sen, että rakkaasta harrastuksestani ei tulisi vain pelkkää työtä.

3.5 Seurantaviikko 19

Maanantai 7.5.2018

Vapaa

Tiistai 8.5.2018

Asetin tiistaille tavoitteeksi saada tehtyä autentikoinnin sovelluksen käyttöliittymään, jotta voisimme alkaa tekemään http-kutsuja ja noutaa oikeaa dataa (tai ainakin lähes oikeaa) palvelimelta. Näennäisesti tehtävän piti olla suhteellisen helppo; minun ei tarvitsisi kuin kopioida suurimmalta osalta toisesta sovelluksesta toimintalogiikka ja muokata muutamaa kohtaa, jotta tiedot vastaavat tämän sovelluksen tietoja. Aamupäivän aikana sainkin tehtyä kaiken tarpeellisen ja jätin testaamisen vasta lounaan jälkeiselle ajalle.

Lounaan jälkeen unohdin kuitenkin koko asian ja siirryin tekemään bugikorjauksia käyttöliittymään. Olin huomannut, että lisäkysymyksissä oleva valintaruutu ei toiminutkaan oi-

kein. Sivulle siirtyessä sen piti olla oletuksena merkitty ja sivulle palattaessa sen piti olla valittuna vain, jos yksi kriteeri täyttyisi, mutta valintaruutu näytti olevan aina valittuna.

Aikani aherrettua valintaruudun kimpussa huomasin, että autentikoinnin osalta jotain oli vielä tekemättä, sillä http-kutsut eivät toimineetkaan vielä, vaikka niin kuvittelinkin. Valitettavasti en ehtinyt tutkia, että missä vika oli, sillä minun piti osallistua statuspalaveriin, jotta voisin kertoa asiakkaalle, miten asiat olivat edenneet. Palaveri kestikin melkein kaksi tuntia, koska jäimme keskustelemaan asiakkaan toimittamista Excel-tiedostoista ja niiden sisältämistä virheistä.

Keskiviikko 9.5.2018

Tavoite keskiviikolle oli sama kuin tiistaille; saada autentikointi toimimaan. Käytin asian selvittelemiseen käytännössä koko päivän, mutta vaikka löysinkin virheitä ja syitä siihen miksi kutsut eivät toimi, en saanut asiaa korjattua.

Autentikointi hoituisi niin, että http-kutsun headerissa lähetetään sekä käyttäjänimi että access token ja ne kutsun yhteydessä lähtivätkin ihan oikein, mutta jostain syystä palvelin yritti edelleen ohjata sisäänkirjautumissivulle. Hypin koko päivän palvelin- ja käyttöliittymäkoodin välillä yrittäen löytää ongelman juurisyytä, mutta tuloksetta.

Päivä ei silti mennyt kokonaan hukkaan sillä poistelin viimeisetkin jämät hakukriteerien sovelluksen muistiin tallentavasta koodista. Nyt yksikään komponentti ei enää hyödynnä sovelluksen omaa muistia, vaan koko haku perustuu URL-osoitteen perässä mukana kulkeviin hakuparametreihin, joten esimerkiksi hakutulossivulle voisi siirtyä pelkällä URL-osoitteella, kunhan parametrit olisivat oikein.

Torstai 10.5.2018

Helatorstai.

Perjantai 11.5.2018

Viikon viimeisen työpäivän tavoitteeksi asetin saada työpisteeni kasattua ja valmiiksi tulevaa viikkoa varten. Meille oli varattu uusi neuvotteluhuone, missä tulisimme olemaan projektin loppuun saakka. Saavuin asiakkaan tiloihin jo ennen kuutta, mutta olin täysin unohtanut sen, että neuvotteluhuoneeseen pääsisi vasta aikaisintaan aamulla yhdeksän aikoihin. Tein siis ensimmäiset kolme tuntia töitä kyseisen kerroksen yleisessä keittiötilassa.

Minun onnekseni moni oli ottanut ilmeisesti perjantain vapaaksi helatorstain kunniaksi ja loput eivät olleet aamuvirkkuja, sillä sain olla lähestulkoon yksin koko kerroksessa aina aamu kahdeksaan asti.

En kuitenkaan istunut toimettomana, vaan jatkoin autentikointiongelmien selvittelyä. Tuijotettuani tunnin verran koodia tuskastuin, koska asia ei tuntunut etenevän ollenkaan. Päätin sysätä ongelman sivuun ja katsoa sitä yhdessä kollegoideni kanssa, kunhan he vain saapuisivat töihin.

Koska en heti keksinyt, että mitä tekisin seuraavaksi enkä nähnyt mitään suurempia bugeja, päätin yrittää löytää ratkaisuntynkää asiakkaan arkkitehtuuriongelmaan. Jonkin aikaa googletettuani muistin työkaverini löytämän videon, joten päätin katsoa sen. Videolla vaateverkkokauppa Zalandolla töissä olevat kehittäjät esittelivät Zalandon arkkitehtuuriratkaisuja. Rehellisesti sanottuna en odottanut videolta paljoakaan, koska en uskonut Zalandon painivan vastaavanlaisien ongelmien kanssa, sillä kyseessä on kuitenkin vain vaatteita verkossa myyvä kauppa. Jo alkumetreillä huomasin olevani todella väärässä. Zalandolla on melkein 2000 työntekijää töissä teknologiaosastolla ja heidän ratkaisunsa arkkitehtuuriongelmiin oli heidän kehittämänsä Projekti Mosaiikki eli Project Mosaic (Project Mosaic 2016). Kokemattomiin silmiini ratkaisu vaikutti todella hyvältä, mutta sen sopivuudesta asiakkaalle voitaisiin olla monta mieltä. Sen käyttöönotto saattaisi tarkoittaa todella työtä jo olemassa olevien sovellusten parissa, jotta ne olisivat sopivia Zalandon kehittämään ratkaisuun.

Kun olimme päässeet meille osoitettuun neuvotteluhuoneeseen ja saaneet niin sanotut työpisteemme kasattua kyselin vähän autentikointiongelmista. Ilmeisesti vika oli siinä, että jossain vaiheessa joku työkavereistani oli laittanut projektin riippuvuudeksi Spring Security ohjelmistokehyksen, mutta sitä ei ollut otettu käyttöön kunnolla. Tämä aiheutti sen, että vaikka http-kutsun mukana lähtivät tarvittavat tiedot autentikointia varten, niitä ei käsitelty ollenkaan vaan sen sijaan vastauksena tuli uudelleenohjaus kirjautumissivulle. Päätimme yhdessä tuumin tehdä sen heti maanantaina.

Sen vähän ajan, mikä minulla oli vielä työpäivää jäljellä käytin työkavereideni kanssa keskustelemiseen siitä, että mikä olisi paras tapa hoitaa PDF-tiedostojen luonti.

Viikkoanalyysi

Tällä viikolla työpäiviä oli vain kolme ja niistä vain yhden pidin täytenä työpäivänä, joten viikko ei ollut kovin tuottoisa. Jälleen kerran sain käyttää paljon aikaa kehitysympäristön

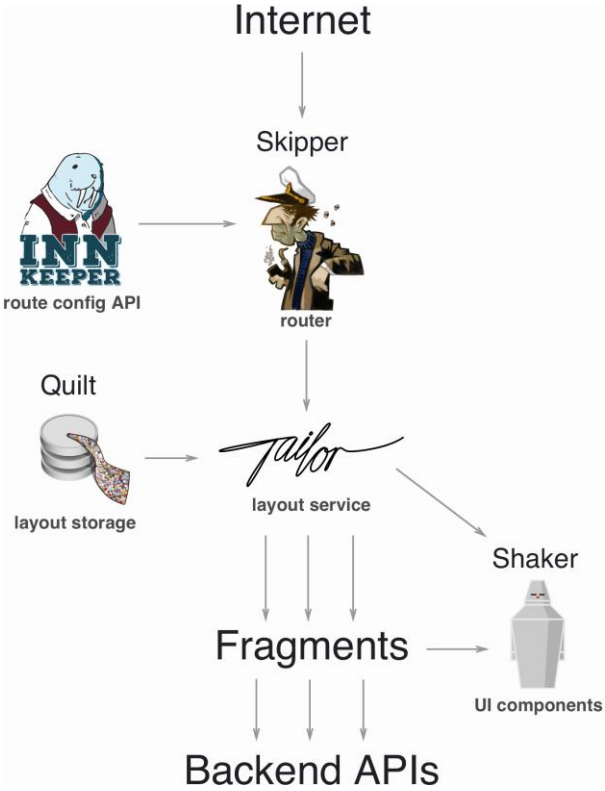
toimivaksi saattamiseen, mikä on alkanut aiheuttaa todella paljon turhautumisen tunnetta. Joinain päivinä tuntuu menevän suurin osa ajasta siihen ja varsinaiselle kehitystyölle jää todella vähän aikaa.

Olen jo useamman kerran pysähtynyt miettimään, että miksi kehitysympäristön toimintaan saaminen on niin vaikeaa ja ennen kaikkea miksi se on niin epävakaa. Voisin kuvitella, että olisi asiakkaan kannalta äärimmäisen tärkeää, että ympäristö on helppo käynnistää ja se olisi vakaa. Pelkästään minulla on mennyt jo varmaan yhteensä kolmen täyden työpäivän verran aikaa hukkaan ja se on hyvin varovainen arvio. Tämä yhdistettynä siihen, että meitä on kuitenkin useampi kehittäjä tässä projektissa ja asiakas maksaa myös jokaisesta niistä tunneista, jotka käytämme kehitysympäristön korjaamiseen ja/tai uudelleen asentamiseen. Kaiken lisäksi kuulin työkavereilta, että samat ongelmat olivat vaivanneet jo edellisen projektin aikana. Tästä voi helposti päätellä, että asiakas on joutunut maksamaan varmaan jo yhteenlaskettuna kuukauden verran siitä, että olemme käytännössä istuneet toimettomina yrittäen saada kehitysympäristöä toimimaan.

Viikon ehdottomasti parasta antia oli katsomani video Zalandon arkkitehtuuriratkaisuista (Fellner & Persa 2017), vaikka en ihan täysin niitä ymmärtänytkään. Parasta siinä oli se, että ne ovat kokonaan open sourcea eli kuka tahansa voi ottaa ne käyttöön ja/tai osallistua kehitykseen. Mosaiikki koostuu kuudesta eri palvelusta, joilla jokaisella on oma tehtävänsä siinä, että verkkosivu latautuu mahdollisimman sujuvasti käyttäjällä ja useat eri tiimit eri mantereilla voivat osallistua kehitykseen ketterästi. Lisäksi tuotantoonvienti on ongelmatonta, kun jokainen osa on itsenäinen, eikä kykene rikkomaan muiden komponenttien toimintaa.

Mosaiikin (Project Mosaic 2016) palvelut ovat lyhykäisyydessään nämä:
Tailor kokoaa verkkosivun saamistaan palasista (fragments), jotka se noutaa asynkronisesti.
Skipper on niin sanottu välittäjä, joka hoitaa http-kutsut oikeisiin paikkoihin ja muokkaa niitä tarvittaessa.
Shaker on kirjasto käyttöliittymäkomponentteja, mitä käytetään Zalandossa, jotta ulkoasu pysyy yhteneväisenä.
Quilt on Tailorin käyttämä sapluunavarasto, mikä tarjoaa rajapintoja useille kehitystiimeille sapluunoiden hallintaa varten.
Innkeeper on rajapintapalvelu, jota Skipper hyödyntää välittäessään http-kutsuja.
Tessellate on palvelimella React-koodia luova palvelu, joka tekee JSON-tiedostoista staattisia HTML-tiedostoja.

Mosaiikin toimintamalli (kuvio 5) karkeasti yksinkertaistettuna menee siten, että Skipper saa http-kutsun selaimelta ja vertaa sitä Innkeeperiltä saamiinsa osoitteisiin. Sen jälkeen Skipper välittää kutsun Tailorille oikeaan paikkaan, joka vuorostaan kasaa kutsun perusteella Quiltilta saamistaan sapluunoista verkkosivun, joka näytetään käyttäjälle (Project Mosaic 2016).



Kuvio 6. Projekti Mosaiikin arkkitehtuurikuvaus (Project Mosaic 2016)

Itselleni jäi kuitenkin epäselväksi, että miten tätä ratkaisuja voisi hyödyntää muidenkin JavaScript-ohjelmistokehysten kuin Reactin kanssa. Videolla kyllä mainittiin, että jokainen tiimi voi valita haluamansa ohjelmistokehyksen. Ilmeisesti se toimisi jotenkin niin, että sapluunat tehtäisiin sillä ohjelmistokehyksellä, millä halutaan, mistä muodostettaisiin JSON-tiedosto, joka lähetetään Tessellatelle HTML-tiedostojen generoimista varten. Asia on kuitenkin siinä määrin mielenkiintoinen, että tulen tästä keskustelemaan työkavereideni kanssa tulevaisuudessa.

Kuviosta 6 on havaittavissa lisäksi, että toiminta näyttää selkeältä, yksinkertaiselta ja aika suoraviivaiselta, mutta voisin kuvitella, että toteutus ei ihan niin helppoa ole kuin mitä kuva antaa ymmärtää. Jos joskus löydän ylimääräistä vapaata aikaa ja olen kartuttanut enemmän kokemusta, voisin yrittää saada tehtyä jonkin oman projektin Mosaicia hyödyntämällä. Uskon, että siinä oppisi paljon uutta ja että siitä saattaisi olla hyötyä tulevaisuudessakin.

3.6 Seurantaviikko 20

Maanantai 14.5.2018

Koska edeltävä viikko oli mitä oli tuottavuuden suhteen, asetin päivälle ronskin tavoitteen, joka oli muokata PDF-tiedostojen luomiseen käytetty testimetodi uuteen uskoon ja sitä myöten myös rajapinta täytyisi tehdä uudelleen. Ryhdyinkin heti aamusta tuumasta toimeen innosta puhkuen ja aloin suunnittelemaan. Ihan ensimmäiseksi lueskelin työkaverini tekemää HTML-sapluunaa ja palautin mieleeni viime perjantain keskustelut. Hetken pähkäiltyäni päätin tehdä kokonaan uuden Java-luokan, jota hyödynnettäisiin PDF:n luomisessa.

Uusi luokka jäi melko suppeaksi; se sai sisällökseen vain muuttujat ”title” ja ”content” ja tietysti etenkin Javalle (tai ehkä paremminkin olio ohjelmoinnille) ominaiset *getterit* ja *setterit* ja tietysti konstruktorin parametreilla ja ilman. Sen jälkeen muokkasin testimetodin uuteen uskoon ja aloin ajamaan testejä. Aamupäivän saldo oli siis hyvä. Olin saanut testit menemään läpi ja tuloksen lähes halutuksi.

Pidimme päivän aikana parin kollegan kanssa palaverin, jossa suunnittelimme rajapintoja. Pääsimme aika nopeasti konsensukseen siitä, mitä tarvitaan kummallakin puolella ja kuka on vastuussa mistäkin.

Päivän aikana poistin koneeltani Java kehitysympäristö Eclipsen, koska sen toiminta oli ollut hyvin vaihtelevaa viime aikoina. Eclipsen tilalle päädyin asentamaan JetBrainsin kehittämän IntelliJ:n Community Editionin, joka on ilmainen. Vaikka kehitysympäristö onkin pääpiirteissään suhteellisen samanlainen kuin Eclipse, niin jouduin kuitenkin käyttämään tunnin verran aikaa siihen, että pääsin jyvälle päätoiminnoista, koska en ollut käyttänyt ohjelmaa aikaisemmin.

Yksi suurimmista ärsytyksen aiheista Javalla (ja muilla kielillä kuin JavaScript) ohjelmoidessa on se, että käyttöliittymästä tulevien JSON-objektien käsittely on kankeaa. Yksi suuri syy tähän on luonnollisestikin se, että Java on staattisesti tyypitetty kieli toisin kuin JavaScript. Tarkoituksena on lähettää PDF:n luomiseen käytettävään rajapintaan kuvion 7 mukainen JSON-objekti ja sen muuttaminen Javalle toimivaan muotoon aiheutti suuria ongelmia itselleni rajapintaa tehdessä. Jouduin tekemään aiemmin tehdylle Java-luokalle niin sanotun wrapper-luokan, jotta saisin logiikan toimimaan edes jotenkuten. Valitettavasti en ehtinyt saada sitä ihan valmiiksi päivän aikana, joten vasta huomenna selviää, toimiiko tekemäni ratkaisu.

Kuvio 7. Esimerkki JSON-objekti

Tiistai 15.5.2018

Asetin päivälle saada PDF-tiedoston generointitestin valmiiksi, sillä alun perin suunnittelemani JSON (kuvio 7) muuttui hieman, jotta se toimisi oikeasti. Kuviossa 8 näkyy uusi muoto. Ongelma oli edelleen sama eli miten muuttaa JSON Javalle sopivaan muotoon? Tuskailin jonkin aikaa ongelman parissa ja lopputulos näytti tältä:

ArrayList<HashMap<String, String>, HashMap<String, ArrayList<HashMap<String, ArrayList<JavaLuokka>>>>

Melko hirveän näköinen. Loppujen lopuksi päädyin kollegan neuvosta tekemään toisen wrapper-luokan. Se ei saanut kuin kaksi muuttujaa, mutta ratkaisi ongelman ja teki lopputuloksesta siistimmän. Luokan muuttujat:

HashMap<String, String> ja ArrayList<JavaLuokka>

```
{
    "category": "category1",
    "answers": [
        {"title": "answer1", "content": "content1"},
        {"title": "answer2", "content": "content2"}
    ],
    "category2": "category2",
    "answers2": [
        {"title": "answer3", "content": "content3"},
        {"title": "answer4", "content": "content4"}
    ]
}
```

Kuvio 8. Uudelleen muotoiltu JSON-objekti

Muuttelin HTML-sapluunan for-silmukat oikeaan muotoon ja sen sekä tämän uuden wrapper-luokan avulla sain testit menemään lävitse ja PDF:t generoitua. Varsinaisen rajapinnan pariin en tänään päässyt, sillä iltapäivällä oli parin tunnin sprintin suunnittelukokous ja kehitysympäristöni lakkasi toimimasta.

Keskiviikko 16.5.2018

Tavoitteen keskiviikolle olin asettanut jo edellisenä päivänä, joka oli edistää PDF:n luomiseen tarkoitettua rajapintaa mahdollisimman paljon.

Aamupäivä meni kuitenkin siihen, että kollegan kanssa laitoimme kehitysympäristöni toimimaan ja sen jälkeen hän esitteli tekemänsä uudet rajapinnat. Asiakkaan toiveena oli, että ottaisimme ne käyttöön mahdollisimman pian, jotta he voisivat esitellä sovelluksen toimintaa oikealla datalla.

Kun kollegani sai rajapinnat esiteltyä ja kerrottua, että miten niitä käytetään, ryhdyin toimeen. Koko loppupäivä meni koodia refaktoroidessa, joka tarkoittaa sitä, että ohjelmakoodia muutetaan selkeämmäksi ilman, että toiminnallisuus muuttuu tai uusia ominaisuuksia lisätään, ja sen jälkeen http-pyyntöjä tehdessä. Oikean datan käyttö vaati yllättävän paljon muutoksia jo olemassa olevaan koodiin, mitä en osannut ennakoida, joten työtä riitti, jopa niin paljon, että sama työ jatkuu vielä huomenna.

Tein näitä muutoksia tehdessä uuden haaran versionhallintaan, minne lopulta tallensin muutokset. Se, miksi en käyttänyt master-haaraa, johtui lähinnä kahdesta syystä, jotka olivat, että koodi ei ollut valmista ja toimivaa ja koska serveripuoli ei ole vielä siinä vaiheessa, että siitä voisi tehdä paketin testiympäristöön käytettäväksi. Tämän takia käyttöliittymäkään ei voi olla siinä tilanteessa, että tarvitsisi backendin koodia toimiakseen. En

varsinaisesti päässyt tavoitteeseeni, sillä en ehtinyt edistää PDF-rajapintaa ollenkaan, mutta ehkä ehdin tehdä sitä huomenna tai viimeistään perjantaina.

Torstai 17.5.2018

Tavoite päivälle oli yksinkertainen eli saada hakutoimintoon liittyvät rajapinnat täysin valmiiksi. Koska tavoite oli niin selkeä, oli helppo ryhtyä toimeen heti kun pääsin töihin. Päivästä ei hirveästi jäänyt kerrottavaa, sillä tein rajapintoja aina siitä hetkestä, kun tulin töihin, siihen asti, että lähdin töistä.

Muutoksia olemassa olevaan koodiin piti tehdä huomattavasti enemmän kuin mitä eilen vielä osasin ennakoidakaan. Yksi ratkaistava ongelma oli se, että kun käyttäjä on valinnut ensimmäiset hakukriteerit ja haku tehdään seuraavalle sivulle siirryttäessä, yksi mahdollinen uusi kysymys on jo etukäteen valittuna. Tästä seurasi, että uutta hakua ei tietenkään tapahtunut, koska käyttäjä ei ollut tehnyt mitään. Käyttöliittymä sen sijaan jäi sellaiseen tilanteeseen, että näytti siltä, että jotain tapahtuu taustalla tai että ohjelma on jäänyt jumiin. Ongelma ei loppujen lopuksi ollutkaan niin haastava kuin millaiseksi sen olin mielessäni muodostanut. Riitti, että tein uuden kyselyn oletusdatalla, jotta käyttäjä pääsee eteenpäin.

Sain tehtyä hakuun liittyvät toiminnot kaikki valmiiksi siten, että sovellusta voi käyttää jo oikeaa dataa vasten ja hakutulokset vastaavat todellisuutta. Työtä on kuitenkin edelleen vielä paljon edessä hakutoimintojen kanssa. Esimerkiksi näkyville tulevat kysymykset eivät tule loogisessa järjestyksessä vaan hyppäävät sinne, missä vapaata tilaa sattuu olemaan. Toinen isompi asia on se, että jos käyttäjä haluaa muuttaa hakuaan, pitäisi kaikki sen kysymyksen jälkeen tehdyt valinnat nollata ja kysymykset poistaa näkyvistä, jotta ei pääse syntymään tilannetta, missä käyttäjä on tehnyt sellaisia valintoja, mille ei löydy vastausta.

Kaiken kaikkiaan olen tyytyväinen päivän saavutuksiin, koska sain tehtyä hakutoiminnallisuuden sen verran valmiiksi, että sitä voidaan tarvittaessa esitellä asiakkaalle. Päivän aikana tuli luettua jälleen kerran Angularin dokumentaatiota jonkin verran, jotta sain kaikki toimimaan halutulla tavalla. Huomasin kuitenkin sen, että osaamiseni on kehittynyt tässä kuluneen kuuden viikon aikana huimasti Angularin käytössä, sillä vaikka dokumentaatiota tuleekin luettua, ei sitä tarvitse lukea lähellekään niin paljon kuin muutama viikko sitten.

Perjantai 18.5.2018

Viikon viimeiselle päivälle asetin suhteellisen kevyen tavoitteen eli viimeistellä haun toiminnallisuus ja korjailla bugeja, mikäli niitä löytyisi. Päätin aloittaa päivän varovaisesti ja korjasin ensimmäiseksi yhteenveto- ja tulossivulle hakuparametrit näkyviin, mitkä olivat hävinneet edellisen päivän aikana http-kutsuja tehdessä. Tämä oli helppo korjaus, mutta todennäköisesti vain väliaikainen sellainen. Koska jouduin muuttamaan sovelluksen koodia niin paljon, ei URL-osoitteen mukana kulkevia hakuparametreja käytännössä käytetä enää ollenkaan ja tämä täytyisi korjata tulevaisuudessa, kunhan siihen liittyviin ongelmiin löytyy ratkaisu.

Seuraavaksi aloitin työt kriittisimmän toiminnon korjaamisella / tekemisellä. Käyttäjän muuttaessa vastauksiaan pitäisi sen jälkeiset vastaukset poistaa ja tehdä uusi haku, jotta käyttöliittymään tulisi näkyviin varmasti oikeat kysymykset. Tällä oli tarkoitus poissulkea sellainen mahdollisuus, että käyttäjä onnistuisi tekemään sellaiset valinnat, mille ei löydy vastausta. Alun perin vähän jännitin, että olisiko tämä kuinka haasteellinen tehtävä, mutta onnistuin kirjoittamaan toimivan ratkaisun yllättävän helposti.

Kun vanhojen vastausten nollaus oli tehty valmiiksi, päätin parantaa yhden hakutuloksen toiminnallisuutta. Ongelmana oli, että jos vastausvaihtoehto oli mallia kategoria ja vastaus eikä pelkkä vastaus, yhteenveto- ja tulossivulle tuli näkyviin pelkkä vastaus ja vastauksen kategoria puuttui kokonaan, joten lopputulos saattoi näyttää hassulta. En valitettavasti keksinyt parempaa ratkaisua siihen kuin kaksi sisäkkäistä silmukkaa, jolla etsitään oikea kategoria ja vastaus - pari. Tähän täytynee palata myöhemmin vielä uudemman kerran.

Pääsin myös edistämään PDF-tiedoston luomiseen tehtyä rajapintaa ja se olikin päivän mielenkiintoisin osuus. Aikaisemmin tekemäni uuden wrapper-luokan toinen muuttuja oli valittu huonosti ja tämä aiheutti pientä päänvaivaa. Parin tunnin touhuamisen jälkeen sain kuitenkin itse rajapinnan toimimaan muuten oikein, mutta valitettavasti se ei vielä osannut palauttaa luotua PDF-tiedostoa käyttöliittymään. Tässä ongelmaksi tuli se, että http-kutsut hoitava controller-luokka ei löytänyt PDF-tiedostoa ja kaatui virheeseen. Valitettavasti en ehtinyt tänään sitä ratkaista, sillä iltapäivä meni muutamia bugikorjauksia tehdessä käyttöliittymään, jotta testiympäristöön saataisiin mahdollisimman toimiva versio sovelluksesta.

Viikkoanalyysi

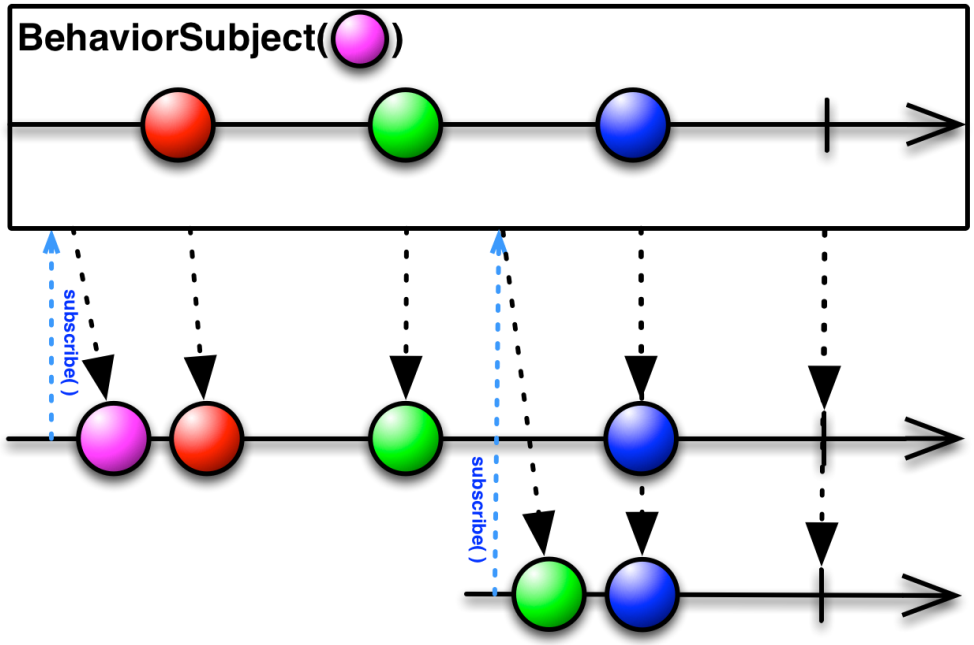
Kulunut viikko oli omalla tavallaan raskas. Pidän kyllä siitä, että on töitä mitä tehdä, etenkin kun pidän työstäni, mutta vaikka tämä työ ei ole fyysisesti raskas, en osannut odottaa kuinka raskasta jatkuva ongelmanratkaisu onkaan. Aikaisemmissa töissäni raskas päivä

on tarkoittanut sitä, että olen fyysisesti väsynyt, mutta nyt väsymys on ihan erilaista eikä siitä toivu lähellekään niin nopeasti kuin fyysisestä rasituksesta.

Uskon, että tilanne muuttuu sitä mukaa, mitä enemmän minulla on kokemusta näistä asioista, koska uusien asioiden opettelu on helpompaa olemassa olevan laajemman tietopohjan ansioista ja samantapaisia ongelmia on tullut ratkottua useampaan kertaan, joten ne eivät rasita enää samalla tavalla.

Tämän viikon parasta antia on ollut loppuviikon työ rajapintojen parissa käyttöliittymäpuolella. Vaikka nautin myös serveripuolen koodin tekemisestä ja se on se asia mihin tulevaisuudessa haluaisin niin sanotusti erikoistua, on sen tekeminen todella paljon stressaavampaa, koska osaamistasoni ei Javassa ole niin hyvä kuin JavaScriptissä. Tämä aiheuttaa sen, että en tavallaan uskalla tehdä mitään, sillä pelkään muiden mahdollista arvostelua ja tässä asiassa minun pitää kehittyä paremmaksi. Pitää uskaltaa tehdä enemmän ja varsinkin tehdä virheitä, sillä niistä oppii parhaiten.

Haun oikean toiminnallisuuden tekeminen oli miellyttävää, mutta haastavaa puuhaa. Ennen kuin ryhdyin kirjoittamaan koodia, piti minun ensin suunnitella, että miten sen tulen toteuttamaan ja mikä olisi paras tapa toteuttaa tämä toiminto. Hakua tehdessä käyttöliittymästä lähtee useampi kutsu samaan osoitteeseen ja vain kyselyn sisältö vaihtuu käyttäjän vastausten mukaan. Päädyin käyttämään BehaviorSubjectia serveriltä saadun vastauksen tallentamiseen. BehaviorSubject oli mielestäni hyvä ratkaisu tähän, koska sen toimintatapa on sellainen, että niin sanottu tarkkailija tilaa (subscribe) BehaviorSubjectin datan ja joka kerta kun se muuttuu, lähtee jokaiselle tilaajalle automaattisesti uusin data. BehaviorSubject palauttaa uusille tilaajille aina tilauksen tapahtumahetkellä olevan datan tai mahdollisen oletusarvon, jos dataa ei vielä ole (kuvio 9). Virheen sattuessa jokaiselle tilaajalle tulee virheviesti ja sama koskee myös uusia tilaajia: jos virhe on tapahtunut ennen kuin tarkkailija tilaa datan, saa hän virheviestin vastineeksi (ReactiveX 2018). BehaviorSubject ei muista vanhaa dataa, vaan siltä voi saada vain uusimman arvon. Käyttöliittymään kysymykset tulevat näkyviin yksitellen hakukriteerien mukaan, mutta kysymykset joihin on vastattu jo, pitää pysyä näkyvissä. Tämä yhdistettynä siihen, että BehavioSubjectilta ei saa vanhaa dataa johti siihen, että jo kysytyt kysymykset piti tallentaa erikseen sitä mukaa, kun niitä saatiin, jotta kaikki kysymykset pysyisivät käyttöliittymässä näkyvillä.



Kuvio 9. BehaviorSubjectin toiminta kuvana (ReactiveX 2018)

Tekemäni hakutoiminto toimii niin, että palvelussa (service), minne kysymystenhakulogiikka on tehty, metodi hakee uuden kysymyksen annetuilla vastauksilla ja tallentaa sen palvelun omaan muuttujaan, jonka komponentit voivat tilata. Päädyin tällaiseen ratkaisuun koska halusin, että ensimmäisen kysymyksen haku tapahtuu heti, kun käyttäjä painaa ensimmäisellä sivulla jatka-nappia. Vastaus piti siis tallentaa johonkin, että seuraava sivu (eri komponentti) saa tietoonsa palvelimen palauttaman vastauksen ilman, että pitäisi tehdä uusi http-kutsu.

Oikeastaan, kun tarkemmin asiaa ajattelen, niin tämä viikko oli kyllä seurantajakson paras ja samalla raskain. Sain pyöritellä koko viikon ajan mielenkiintoisia ongelmia mielessäni ja etsiä niihin ratkaisuja. Osaamisen Javassa (ja miksei toki uudessa kehitysympäristössä IntelliJ:ssä) sekä Angularissa kehittyi huimasti. Joka viikko Angular tuntuu entistä luontevammalta ja tätä on tukenut varmasti paljon se, että harjoittelen sitä myös vapaa-aikani omassa projektissani.

Yksi iso asia, joka tästä viikosta on tehnyt raskaan, on lähestyvä deadline demolle. Varsinkin, kun asiakas on koko ajan nostanut vaatimuksia, että mitä demossa pitäisi olla. Alun perin vain sovelluksen käyttöliittymän tulossivulla piti olla kovakoodatut arvot tietyille hakukriteereille. Seuraavaksi vaatimuksena oli, että sovelluksesta pitäisi saada myös PDF-tiedosto samoilla tiedoilla ulos. Tällä viikolla vaatimukseksi tulikin täysin toimiva haku oikealla datalla. Jos kollegoiden puheista luin rivien välistä oikein, oli meno samanlaista edellisessä projektissa. He puhuivatkin leikkisästi demo driven developmentista, eli kehitystyö tapahtuu demon vaatimien toiminnallisuuksien tekemisessä eikä välttämättä loogi-

sessa järjestyksessä, miten alun perin oli suunniteltu. Kuulin myös loppuviikosta asiakkaan puolelta vihjailua, että viikonloppunakin voisi tehdä töitä projektin parissa. En ole ihan varma, mitä mieltä siitä olen. Voisin ihan hyvin tehdäkin viikonloppuna töitä, mutta taas toisaalta on minulla muutakin tekemistä ja kyse ei ole varsinaisesti ylityöstä, vaan tehdyt tunnit kerryttäisivät saldoa. Ei erillistä viikonloppukorvausta eikä ylityölisää.

Tämä muistutti minua Engin blogikirjoituksesta, jonka olin lukenut joskus viime vuonna ensimmäisen kerran. Kirjoituksessa hän kertoo käyneensä läpi burnoutin 1990-luvun loppupuolella ja syiksi hän listasi muun muassa tiukat deadlinet, pitkät työpäivät ja työt viikonloppuina (Eng 2017). Itselläni on ikää jo sen verran, että muutakin elämää on ehtinyt karttua työn ympärille, joten viikonlopuista harvemmin edes löytyisi oikeasti aikaa työnteolle, vaikka halua olisikin. Ehkä se on hyvä asia, koska itseni tuntien saattaisin helposti uhrata illat ja viikonloput työlle, jos siihen olisi mahdollisuus, kuten jo pari viikkoa sitten totesin sen viikon viikkoanalyysissä. Mutta vaikka elämäntilanteeni ei sallikaan jatkuvaa työntekoa, en näe sitä täysin mahdottomana, että kokisin burnoutin samaan tapaan kuin Eng. Huomasin jo nyt stressitasoni nousevan jonkin verran, vaikka en kovin helposti stressaannukaan, kun asiakas jatkuvasti muistutti tulevasta demosta ja vaatimukset vain lisääntyivät. Huolestuttavinta tässä on se, että nyt on kuitenkin vasta kyse demoversiosta, joten jään mielenkiinnolla odottamaan, mitä se on, kun sovellus pitäisi viedä tuotantoon.

3.7 Seurantaviikko 21

Maanantai 21.5.2018

Päivän tavoite oli kaikessa yksinkertaisuudessaan se, että halusin saada tuon jo surullisen kuuluisan PDF-tiedoston luomiseen tarkoitetun rajapinnan toimimaan täysin. Se oli roikkunut mukana jo liian kauan keskeneräisenä. Aloitin ongelman purkamisen ensin selvittämällä, että miksi controller-luokka ei löytänyt tiedostoa. Kokeilin todella monia eri vaihtoehtoja tiedoston avaamiseen ja lopulta onnistuin löytämään oikean ratkaisun. Koodi, jolla sain sen toimimaan, näytti tältä:

```
ClassLoader classLoader = new Thread.currentThread().getContextClassLoader();
File file = new File(classLoader.getResource(“tiedosto.pdf”).getFile());
```

Sen jälkeen kaikki toimi niin kuin pitkin. Tai ainakin luulin niin. Jonkin aikaa rajapintaa testailtuani huomasin, että tiedosto, joka palautetaan, on koko ajan niin sanotusti yhden jäljessä. Ensimmäisellä kerralla palautui oikea tiedosto, mutta kun rajapintaa kutsui uudemman kerran, sai vastauksena ensimmäisen kutsun yhteydessä luodun PDF:n ja kolman-

nella kutsulla sai toisella kerralla luodun PDF:n ja niin edelleen. Yritin ratkaista ongelmaa omin voimin parin tunnin verran tuloksetta ja sitten oli pakko pyytää kollegalta apua. Hänen ehdotuksensa oli, että tiedostoa ei ikinä edes tallennettaisi tiedostojärjestelmään, koska siitä voi aiheutua muitakin ongelmia. Sen sijaan se luettaisiin muistiin siksi aikaa, kunnes se on palautettu käyttäjälle. Sain muutettua rajapinnan toiminnan työkaverini ehdotuksen mukaiseksi suhteellisen nopeasti ja se poisti kaikki ongelmat PDF:n luomiseen ja käyttäjälle palauttamiseen liittyen.

Seuraava askel olikin saada käyttöliittymä lähettämään tietoa oikeassa muodossa rajapintaan ja vastaanottamaan luotu PDF. Tämä olikin nopeasti hoidettu, koska pystyin katsomaan mallia työkaverini tekemästä vastaavanlaisesta ratkaisusta edellisestä projektista. Ainoa ongelma, jota jouduin hetken aikaa miettimään, oli se, että olin epähuomiossa laittanut http-kutsun odottamaan vastausta, joka olisi sisältötyypiltään ’application/octet-stream’, kun sen piti olla ’application/pdf’ ja tämä aiheutti sen, että käyttöliittymä ei osannut käsitellä palvelimelta saatua vastausta oikein.

Koska rajapinta näytti toimivan täysin oikein, päätin seuraavaksi paneutua seuraavaan ongelmaan PDF:ään liittyen. Koska PDF-tiedoston sisältö tuotettiin dynaamisesti for-silmukan avulla, sivunvaihdot eivät menneet oikein ja vastauslaatikko saattoi jakautua kahdelle eri sivulle, mikä ei ollut toivottava tilanne. En lähtenyt ratkomaan tätä ongelmaa kovin luottavaisin mielin, sillä kollegani oli paininut saman ongelman parissa ainakin päivän verran tuloksetta. Etsin internetistä vartin verran erilaisilla hakutermeillä ja lopulta osuin kultasuoneen. Ongelman ratkaisu oli harmillisen yksinkertainen ja olisihan meidän pitänyt tajuta se itsekin, sillä kaikki mitä vaadittiin, oli ’page-break-inside: avoid;’ -tyylin lisääminen haluttuun elementtiin. En oikein tiennyt olisiko pitänyt itkeä vai nauraa.

Vielä päivän lopussa vastaan tuli yksi ikävämpi ongelma. Testiympäristössä luotuun PDF:ään ei tyylit eivätkä kuvat latautuneet ollenkaan. Käytin tämän ongelman selvittämiseen parin tunnin verran aikaa ja lopulta löysinkin siihen ratkaisun. Täytyy kyllä sanoa, että Javan tiedostonkäsittely tai tiedostopolkujen käsittely on todella mystinen tapaus ja epäjohdonmukainen. Tai ainakin siltä se tuntuu. Se, mikä toimii omalla koneella kehitysympäristössä ajettuna, ei toimi kun sovellus paketoidaan JAR-paketiksi. Ehkä syy tähän vielä valkenee minulle joskus.

Tiistai 22.5.2018

Asetin tämän päivän tavoitteeksi aloittaa ja saada tehtyä mahdollisimman paljon olemassa olevan ohjelmakoodin parantelua ja ennen kaikkea muuttaa sitä dynaamisemmaksi. Tällä

hetkellä liian moni arvo on kovakoodaattuna käyttöliittymän lähdekoodiin, mikä on todella huono asia. Tämä johtuu pääosin siitä, että jouduimme tekemään käyttöliittymää niin pitkään ilman oikeaa dataa. Tiesin jo ennen kuin aloitin, että edessä on iso urakka, mutta se olisi joka tapauksessa tehtävä, sillä mitä pidemmälle projekti etenee, sitä vaikeammaksi se muuttuisi.

Aloitin refaktoroinnin muuttamalla ensin lisäkysymyssivulta joka ikisen mahdollisen kysymyksen täysin dynaamiseksi ja tein käyttöliittymästä niin sanotusti tyhmän. Tällä tarkoitan sitä, että käyttöliittymällä ei ole mitään tietoa, mikä kysymys on ja jokaista kysymystä kohdellaan samalla tavalla. Kysymysten varsinainen käsittely tapahtuu serveripuolen koodissa. Lähestymistavaksi otin tässä sellaisen, että jokainen kysymysobjekti tallennetaan niin sanottuun kysymystaulukkoon (array), mikä puolestaan käydään läpi Angularin HTML-templatessa silmukalla ja HTML-elementit luodaan dynaamisesti. Sain toteutettua kaiken muun paitsi sen, että jos jo vastattuun kysymykseen vastaa uudelleen, ei käyttöliittymään noudeta uusia tietoja ja käyttäjä voi antaa vääriä vastauksia.

Kun lisäkysymyssivu oli toiminnallisuudeltaan kunnossa, siirryin yhteenveto- ja tulossivujen kimppuun. Nämä olivat loppujen lopuksi todella helppoja ja käytin sisällön luomiseen samaa tapaa kuin lisäkysymyssivulla.

Onnistuin saavuttamaan tavoitteeni ja oikeastaan vielä vähän enemmänkin. En uskonut, että saisin tehtyä yhden päivän aikana näin paljon ja näin isoja muutoksia. Suurimmat ongelmat, mitä muutoksia tehdessä ilmeni, liittyi ulkoasuun ja niiden ratkaiseminen jää loppuviikolle joko työkaverilleni tai sitten itselleni.

Keskiviikko 23.5.2018

Päivän tavoitteena oli saada eilen aloittamani refaktorointi, jos ei nyt ihan valmiiksi, niin ainakin siihen pisteeseen, että tätä varten luomani haara versionhallintaan voitaisiin yhdistää master-haaraan ja viedä testiympäristöön asiakasviraston työntekijöille testattavaksi ja arvosteltavaksi.

Aloitin päivän työt korjaamalla vastausten tarkastusmetodin, joka hajosi eilen. Metodin tarkoituksena on käydä sekä käyttäjältä kysytyt kysymykset että käyttäjän antamat vastaukset läpi ja tarkastaa onko kysymykseen annetun vastauksen arvo muuttunut. Tämä osoittautui paljon suuritöisemmäksi urakaksi kuin aluksi kuvittelin. Varsinaisen metodin korjaamisen jälkeen siirryin tekemään muita muutoksia, mutta pitkin päivää huomasin aina uusia bugeja tässä metodissa, joten sain korjailla sitä useita kertoja.

Korjasin myös ne muutamat asiakkaan ilmoittamat bugit, jotka vastaussivulta löytyivät ja samalla varmistin, että ne eivät vaikuttaisi PDF:n luomiseen. Lisäksi sain vihdoinkin tehtyä tarpeettomaksi vanhat metodit ja luokat kaikista muista paitsi yhdestä komponentista. Onnistuin myös korjaamaan käyttöliittymän ulkoasun virheet, mihin olin hyvin tyytyväinen.

Päivä oli hyvin tuottoisa ja saavutin tavoitteeni mainiosti, sillä iltapäivällä kaikki oli valmista master-haaraan sulauttamiseen. Tänään ja eilen tekemäni refaktorointi on ollut opettavaista aikaa ja ymmärrykseni Angularin ja JavaScriptin suhteen on syventynyt todella paljon.

Torstai 24.5.2018

Alan kuulostaa rikkinäiseltä levyltä, mutta päivän tavoite oli saattaa PDF:n generoiminen täysin valmiiksi. Työkaverini yritti eilen ratkaista yhtä siinä ilmennyttä ongelmaa, joka liittyi siihen, että vastausten otsikon kuva ei tullut näkyviin. Vika oli jälleen kerran Javan vähintäänkin mielenkiintoisessa tavassa käsitellä tiedostoja ja niiden polkuja. Ongelma oli kuitenkin vähän haasteellisempi kuin toissapäiväinen vastaava tapaus. Koska sisältö generoitiin for-silmukalla, ei luonnollisestikaan jokaiselle kuvalle voinut käyttää samaa polkua. Tiedostopolun, joka oli muuttuja, sisälle piti saada lisättyä toinen muuttuja, mutta se oli haasteellisempaa kuin miltä kuulostaa. Ikävintä oli, että koska tiedostopolku määriteltiin HTML-sapluunassa, josta generoitiin PDF-tiedosto, oli hankalaa saada kuvan tiedostopolku näkyviin ongelmanselvitystä varten. Lopulta tajusin, että voisin käyttää siihen Thymeleafin th:text -attribuuttia, jonka arvoksi laitoin saman muuttujan, jota käytin kuvan tiedostopolkuun. Tällä tavalla sentään näin, että mikä oli se polku, joka kuvalle tuli PDF:n generoimisen yhteydessä. Tämä oli pieni läpimurto ongelmassa, sillä sen avulla huomasin, että kollegani oli laittanut kuvien nimeen välilyönnin ja sen takia kuvat eivät tulleet näkyviin, vaikka tiedostopolku oli näennäisesti oikein. Muutin tiedostonimet ja lisäsin vielä Thymeleafin sapluunaan paikallisen muuttujan, jossa hakutuloksen otsikko muutettiin vastaamaan kuvan nimeä ja sen jälkeen kaikki olikin kunnossa. PDF-tiedosto oli vihdoinkin sen näköinen kuin pitikin ja päivän tavoite oli saavutettu.

Päivän muihin töihin lukeutui frontendin viimeisestä komponentista turhan koodin poistaminen ja sitä kautta sain poistaa myös yhden luokkamäärittelyn. Lisäsin käyttöliittymään myös paikan uudelle vastaukselle, joka tuli uutena vaatimuksena asiakkaalta edellisenä päivänä.

Isoin työ päivän aikana oli kuitenkin hakuparametrien käyttöönotto uudelleen. Ne eivät olleet enää käytössä kahden edellisen päivän refaktoroinnin takia. Loppujen lopuksi

homma oli helppo, mutta työläs, sillä muutoksia piti tehdä moniin paikkoihin. Ihan valmiiksi en tätä kuitenkaan saanut tehtyä, koska täysi toiminnallisuus vaatisi myös muutoksia backendiin.

Perjantai 25.5.2018

Päivän tavoitteeksi asetin korjata mahdollisimman paljon virheitä lähdekoodissa. En asettanut päivälle mitään sen suurempaa tavoitetta, koska tiesin, että työpäivästä tulisi lyhyt.

Saavutin kyllä tavoitteeni, sillä sain tehtyä useampia bugikorjauksia koodiin, joka liittyi hakuparametreihin ja niiden käyttöön. Oikeastaan sen enempää ei päivästä ole kerrottavana, koska työtunteja ei kertynyt päivälle kuin kolme muista menoista johtuen.

Viikkoanalyysi

Seurantaviikko 21 oli hyvin työntäyteinen ja kiireinen. Onneksi aikataulua ei kiristänyt entisestään asiakkaan halu esitellä sovellusta omille asiakkailleen. Sovellusesittelyä kuluneella viikolla oli useampaankin otteeseen, mutta tällä kertaa esittelytilaisuuteen ei vaadittu uusia ominaisuuksia sovellukseen, mitä haluttaisiin esitellä. Tästä johtuen pystyimme keskittymään rauhassa sovelluksen kehitystyöhön ja ainakin omalta osaltani olen todella tyytyväinen viikon saavutuksiin. Java-osaamiseni kehittyi viikon aikana paljon, sillä olin paljon tekemisessä backendin ohjelmakoodin kanssa ja suurin osa tekemistäni asioista oli minulle uusia.

Viikon suurimmat haasteet ja onnistumisen tunteet ovat vahvasti sidoksissa tekemääni PDF-tiedoston luomisen rajapintaan sekä itse PDF-tiedoston luomiseen liittyviin asioihin. Käytin rajapintaan ja sen toiminnallisuuteen paljon aikaa ja se, että sain sen lopulta toimimaan vaatimusten mukaan, antoi itsevarmuutta ja siitä saatu onnistumisen tunne auttaa jaksamaan jatkossakin vaikeiden ongelmien parissa. Käytin rajapintaa tehdessäni paljon aikaa Thymeleafin dokumentaation lukemiseen sekä avun etsimiseen internetistä.

Thymeleafin dokumentaatio on hyvin kattava, mutta olisin kaivannut sinne lisää tietoa tiedostopolkujen käytöstä – varsinkin silloin, kun sovellus paketoidaan JAR-paketiksi. Dokumentaatiossa kyllä listataan useita tapoja, millä tiedostopolut voidaan merkitä, kuten esimerkiksi absoluuttinen tiedostopolku, sivuun suhteellinen tiedostopolku ja kontekstiin suhteellinen tiedostopolku (Thymeleaf 2018). Valitettavasti yksikään dokumentaation listaamista tavoista ei oman kokemukseni toimi, kun sovelluksesta tehdään JAR-paketti. Tai ei ainakaan silloin, kun Thymeleafilla tehdystä HTML-tiedostosta luodaan PDF-tiedosto. On-

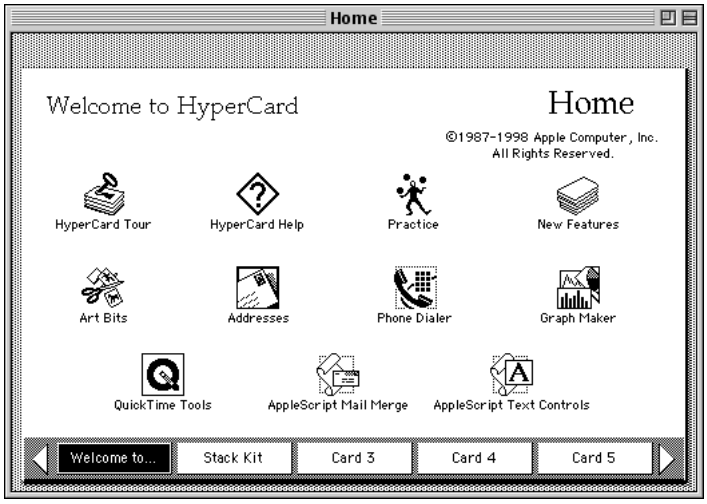
gelman kanssa tuskaillessani tulin huomanneeksi, että en ollut yksin vaan monilla muillakin oli samankaltaisia ongelmia. Muutoin olen kyllä pitänyt Thymeleafista ja voisin kuvitella käyttäväni sitä omassa ohjelmointiprojektissa, jos en haluaisi käyttää käyttöliittymässä mitään JavaScript ohjelmistokehystä. Thymeleaf tuntuu todella monipuoliselta työkalulta ja dokumentaation taso herättää luottamusta.

Viikon aikana sain myös tehtyä sovelluksen frontendissä todella paljon tarpeellisia muutoksia. Suurin osa sovelluksen elementeistä luodaan nyt dynaamisesti sen datan perusteella, mitä palvelinpuolelta saadaan, mikä on erittäin hyvä asia jatkokehityksen kannalta. Dynaamisuus tekee sovelluksesta paljon joustavamman ja kun asiakkaalta tulee lisää vaatimuksia esimerkiksi hakuun liittyen uusien kysymysten muodossa, eivät nämä muutokset välttämättä vaadi ollenkaan käyttöliittymän muokkaamista. Vaatimuksia tulee varmasti vielä lisää, koska asiakas ei ole ehtinyt tehdä kaikkia tarvittavia Exceleitä, mitkä tulevat vaikuttamaan sovelluksen toimintaan, koska kyseiset tiedostot ovat sovelluksen bisneslogiikan takana. Niitä lukemalla päätetään, mitä käyttäjältä kysytään milloinkin ja niistä myös luetaan kyselyn tulokset.

Olen huomannut etenkin tällä viikolla, että mitä pidemmälle sovelluksen kehityksessä etenemme ja mitä valmiimmalta sovellus näyttää, sitä vaikeampi minun on motivoitua tekemään uusia ominaisuuksia tai muutoksia sovellukseen. Huomaan usein ajattelevan, että jokin tietty asia on jo tarpeeksi hyvä eikä kaipaa enää muutoksia tai koko sovellus on jo tarpeeksi hyvä, eikä lisäominaisuuksia enää tarvitse lisätä. Tämä on hyvin huolestuttavaa, varsinkin kun sovellus on selvästi keskeneräinen ja siitä puuttuu paljon ominaisuuksia, mitä vaatimusmäärittelyssä on listattu. En osaa sanoa, että mistä tämä tarpeeksi hyvä -tunne edes kumpuaa, mutta sama niin sanottu ongelma on noussut esiin myös omissa projekteissani. Pahimmillaan se saattaa johtaa siihen, että projekti ei koskaan tule täydellisesti valmiiksi. Asiakasprojektissa työskennellessä on kuitenkin mahdotonta luistaa ja asiakkaan vaatimukset tulee täyttää eikä omalla mielipiteellä ole välttämättä juurikaan väliä. Tietysti osaava ohjelmistokehittäjä kykenee myös kertomaan asiakkaalle, että mitkä ominaisuudet ovat joko täysin turhia, mahdottomia toteuttaa tai liian kalliita toteutettavaksi, mutta itse en ole siihen vielä kykenevä.

Mutta koska tarpeeksi hyvä on tarpeeksi hyvä? Näennäisesti se on helppo määritellä – tehdään vaatimusmäärittely, jonka jälkeen voidaan todeta, että sovellus on valmis kun vaatimukset on täytetty. Todellisuudessa asia ei kuitenkaan ole näin yksinkertainen, sillä jo yhden vajaan projektin perusteella pystyn sanomaan, että vaatimukset eivät ole kiveen hakattuja vaan ne voivat elää todella paljon projektin aikana. Ehkä sovellus on tarpeeksi hyvä kun lähes kaikki bugit on onnistuttu korjaamaan? Tämänkin toteutus on hyvin vaike-

aa, sillä useat bugit löytyvät vasta silloin, kun sovellus on sen loppukäyttäjillä käytössä. Bugien korkea määrä ei suoraan tarkoita sitä, että sovellus ei olisi tarpeeksi hyvä julkaistavaksi. Bachin mukaan Apple julkaisi ensimmäisen version hypermediasovellus ja sovelluskehitysympäristö HyperCardin (kuvio 10), vaikka siinä tiedettiin olevan ainakin 500 bugia ja Macintosh System 7.0 julkaistiin, vaikka siinä oli tuhansia bugeja. Hänen mukaansa kehitystiimit osasivat valita niin sanotusti oikeat bugit julkaisuun mukaan. Kirjoittajan mukaan yksi mahdollinen ratkaisu kysymykseen, että koska vaadittu laatu on saavutettu, on silloin, kun ne ongelmat, mitkä ohjelman pitää ratkaista, on ratkaistu (Bach 2003). Tämä on mielestäni hyvä mittari, mutta siinä on samat ongelmat kuin vaatimusten täyttämisessä eli ongelmat voivat muuttua tai niitä voidaan esittää lisää. Yksi vaihtoehto olisi tietysti se, että vaatimusmäärittelyssä päätetään niin sanotut pääongelmat, joita ei muuteta sovelluskehityksen aikana, jotka sovelluksen täytyy ratkaista ja kun ne on ratkaistu, on sovellus riittävän hyvä ja laadukas julkaistavaksi.



Kuvio 10. Applen HyperCard sovellus (Wikipedia 2009)

Työssäni asiakas on kuitenkin se, joka tekee lopullisen päätöksen sen suhteen, että koska sovellus on tarpeeksi laadukas julkaistavaksi. Tästä voi yrittää neuvotella asiakkaan kanssa, jos itse on täysin eri mieltä ja esimerkiksi sovellus on ihan liian keskeneräinen julkaistavaksi. Joskus tilanne voi kuitenkin olla se, että sovellus julkaistaan, vaikka keskeneräisenä, jos asiakkaan omat aikataulut tai resurssit eivät mahdollista lisäkehitystä.

3.8 Seurantaviikko 22

Maanantai 28.5.2018

Maanantain tavoitteena oli saada valmiiksi hakuparametrien käyttöönotto kokonaisuudessaan. Tämä vaati uuden rajapinnan palvelinpuolelle, josta käyttöliittymään saisi haettua

kysyttyjen kysymysten nimet vastausten perusteella. Kerroin työkaverilleni tästä uudesta tarpeesta ja hän toteutti sen hetkessä, joten pääsin tekemään toiminnallisuutta frontendiin. Hakuparametrien oli tarkoitus toimia siten, että jos käyttäjä päivittää sivun tai tulee sivulle niin, että URL-osoitteen perässä on hakuparametrit, saataisiin palvelimelta haettua kysymysten nimet ja mahdolliset vastaukset. Tällä hetkellä tilanne oli se, että jos käyttäjä päivitti joko yhteenveto- tai tulossivun, suurin osa annetuista tiedoista hävisi, koska käyttöliittymällä ei ollut tietoa kysytyistä kysymyksistä dynaamisuuden takia. Hakuparametreissa oli kysymysten nimien tilalta vain niiden numerotunniste.

Urakka oli hyvä aloittaa tekemällä ensin http-kutsu palvelimelle, jotta tarvittavat tiedot olisi käytettävissä. Toteutin lisäkysymyssivulla tämän niin, että jos hakuparametreja on enemmän kuin neljä (neljän ensimmäisen tiedot saadaan selville muutenkin) ja käyttöliittymällä ei ole tallennettuna kysyttyjä kysymyksiä, tehdään tarvittava http-kutsu. Muussa tapauksessa edetään normaaliin tapaan.

Kun olin saanut kaiken toimimaan niin kuin halusinkin, eli valintalaatikkoihin tuli oikeat arvot valituiksi, oli aika testailla sovellusta. Testejä tehdessä huomasinkin useita ohjelmavirheitä hakuparametrien hyödyntämisessä. Yhteenvetosivulta lisäkysymyssivulle palattaessa arvot olivat oikein, mutta jos vastauksia muutti ja meni yhteenvetosivulle, välittyivät sinne väärät arvot. En onnistunut saamaan ongelman juurisyytä selville ennen päivän päättymistä eli ainakin tiedän, mistä huomenna aloittaa työt.

Tänään pysähdyin hetkeksi ajattelemaan, että kuinka paljon osaamiseni Angular ohjelmistokehyksessä onkaan kasvanut. Olen joka päivä itsevarmempi ja en epäröi tarttua uusiin pulmiin niin paljon kuin alussa.

Trautman kuvaa kirjoituksessaan hyvin sen matkan, joka on edessä kun aloittaa harjoittelemaan ohjelmointia. Hän listaa 4 vaihetta, jotka jokainen käy läpi matkallaan ammattilaiseksi. Nämä vaiheet ovat vapaasti suomennettuna kuherruskuukausi, hämmennyksen kuilu, epätoivon aavikko ja mahtavuuden nousukausi. Kuherruskuukauden aikana löytyy paljon erilaisia resursseja, joiden avulla opetella ohjelmointia ja hämmennyksen kuilu on se vaihe, kun aloitteleva ohjelmoija tajuaa, että ohjelmointi onkin huomattavasti hankalampaa, kuin mitä alussa kuvitteli. Ongelmiksi nousee muun muassa tunne siitä, että ei osaa itse tehdä mitään ja se, että ei osaa kysyä oikeita kysymyksiä. Epätoivon aavikko on pitkä ja yksinäinen matka, kun ohjelmoija yrittää pärjätä vähillä resursseilla ja hioa taitojaan. Lopulta kuitenkin edessä häämöttää mahtavuuden nousukausi, jolloin vihdoin ymmärtää, kuinka sovelluksia oikeasti ohjelmoidaan, vaikka kirjoitettu koodi ei olekaan tuo-

tantovalmista ja ohjelmoijalla ei ole täyttä ymmärrystä kirjoittamastaan koodista (Trautman 2015).
Trautman kirjoittaa hauskasti, mutta osuvasti melkein kuin suoraan minun matkastani työelämään aloittelevasta ohjelmoijasta. Alkuun kaikki oli todella helppoa ja internet tuntui olevan täynnä apua ja sopivia materiaaleja, mutta hämmennyksen kuilu tuli nopeasti vastaan ja oli vähällä, että en luovuttanut ohjelmoimisen suhteen kokonaan. Edessä oli kuitenkin vielä epätoivon aavikko, joka tuntui loputtomalta. Yritin jatkuvasti etsiä apuja internetistä milloin mihinkin ongelmaan ja suurimman osan ajasta tuntui siltä, että en kehity ollenkaan. Nyt olen päässyt jo viimeiseen vaiheeseen ja ohjelmointi tuntuu joka päivä aina vähän helpommalta ja nautin siitä aina vähän enemmän kuin edellisenä päivänä.

Tiistai 29.5.2018

Päivän tavoite oli saada hakuparametrien käyttö sovelluksessa toimimaan kunnolla. Aloitin korjaamalla eilen löytämäni bugin, joka johtui siitä, että arvot otettiin sekä hakuparametreista että sovelluksen muistista. Päätin muuttaa sovelluksen toiminnan kokonaan URL-osoitteessa mukana kulkevien hakuparametrien varaan.

Poistin kaikki turhaksi katsomani toiminnot ja muuttelin ohjelmakoodia sellaiseksi, että sain jokaisen sivun hakemaan tarvittavat tiedot hakuparametreista. Samalla tein myös sellaisen muutoksen, että jokaisella sivulla tehdään uusi http-kutsu palvelimelle, josta puuttuvat tiedot haetaan. Tämä yksinkertaisti sovelluksen rakennetta paljon ja uskon, että samalla se teki sovelluksesta hieman helpommin hallittavan, kun dataa ei yritetä etsiä useasta eri paikasta.

Loppupäivän käytin löytyneiden bugien korjaamiseen, joita tuntui olevan melko paljon muutoksista johtuen.

Tänään asiakas kysyi minulta, että olisinko kiinnostunut jatkamaan heillä toisen projektin parissa tämän nykyisen jälkeen. Olin todella otettu, sillä tämä tarkoitti sitä, että minusta on pidetty ja olen vastannut heidän odotuksiaan. Tämä toinen projekti on se sama, jota projektiryhmässä mukana olevat kollegani tekivät aikaisemmin. Siinä olisi syksyllä tarkoitus aloittaa jatkokehitys ja saada uusi versio julkaistuksi loppuvuodesta. Positiivista asiassa on myös se, että saan jatkaa samojen teknologioiden parissa, joten voin syventää osaamistani sen sijaan, että joutuisin opetella uusia teknologioita seuraavassa projektissa. Mielestäni se on pitkällä tähtäimellä parempi asia, koska näin voin kehittyä vakavasti otettavaksi sovelluskehittäjäksi rauhassa, jolloin uusien teknologioiden opettelu on helpompaa. Lisäksi tulen olemaan tekemisissä jonkun muun kirjoittaman koodin kanssa, joka on

kuitenkin tutun sovelluskehittäjän tekemää, joten sen ymmärtäminen pitäisi olla helpompaa, kuin täysin tuntemattoman ihmisen tekemä koodi, koska olen työskennellyt hänen kanssaan tässä meneillään olevassa projektissa. Ainoa huono puoli asiassa on se, että mahdollisesti meitä ei jää kuin kaksi siihen projektiin. Olen kuitenkin nauttinut näiden henkilöiden kanssa työskentelystä ja saanut heistä kavereita.

Keskiviikko 30.5.2018

Keskiviikon tavoitteeksi asetin itselleni käyttöliittymän toiminnan hiomisen. Käytännössä tämä tarkoitti sitä, että tavoitteena oli saada poistettua siitä mahdollisimman monta bugia.

Päivän mittaan löysinkin paljon asioita, joita korjata ja olen tyytyväinen päivän saldoon. Onnistuin myös päivän aikana löytämään niin sanotusti turhaa koodia ja vajavaista toiminnallisuutta tulossivulta. Sinne oli nimittäin jäänyt muutama arvo kovakoodattuna, joten ne täytyi vaihtaa pois. Työ oli kuitenkin helppo, sillä pystyin hyödyntämään yhteenvetosivulle tehtyä valmista ratkaisua. Nyt koko sovelluksen käyttöliittymä on lähestulkoon täysin dynaaminen ja käyttöliittymällä on käytännössä enää alun kahdesta ensimmäisestä kysymyksestä tieto. Tätä kirjoittaessa tajusin juuri keinon, jolla niistäkin voitaisiin tehdä dynaamisia, jotta käyttöliittymä sopeutuisi mahdollisimman hyvin muutoksiin bisneslogiikassa.

Torstai 31.5.2018

Päivän tavoitteena oli saada tehtyä backendiin sellainen muutos, että tietyille palvelimelta käyttöliittymään lähetettäville arvoille lisättäisiin yksi attribuutti. Tämä johtui siitä, että lista, jossa nämä näytettiin, oli melko kapea ja kaikkien arvojen nimi ei mahtunut yhdelle riville, vaan jakautui kahdelle, mikä ei ollut kivan näköistä.

Ennen kuin tein mitään muutoksia palvelinpuolelle, testasin visiotani tekemällä muutoksen käyttöliittymään, jotta voisin todeta sen toimivuuden. Metodi oli yksinkertainen, sillä siinä käytiin for-silmukalla arvot läpi ja jos nimi oli yli kahdeksan merkkiä tai enemmän, leikattiin se kuudennen merkin kohdalta poikki ja lisättiin loppuun kaksi pistettä. Ratkaisu oli toimiva, joten päätin tehdä sen backendiin. Vaikka asian olisi hyvin voinut jättää frontendin hoidettavaksi, on parempi tehdä se jo palvelinpuolen koodissa, koska siellä ei tarvitse käydä arvoja silmukalla läpi, vaan muutos voidaan tehdä jo arvoja listaan lisättäessä. Lisäksi, jos käyttöliittymässä JavaScriptillä käydään lista lävitse silmukalla, sovelluksen kaikki muu toiminta käyttöliittymässä pysähtyy siksi aikaa, kunnes silmukka on valmis.

Nyt kun käyttöliittymään palautui sopivaa dataa, piti tehdä myös niin sanottu tooltip eli infolaatikko, joka tulee esiin kun hiiren vie muuttujan päälle. Halusin laittaa tähän infolaatikkoon näkyviin muuttujan koko nimen, jotta käyttäjä näkisi sen tarvittaessa. Tähän olisin joutunut todennäköisesti käyttämään paljon aikaa, ellen olisi löytänyt valmista ja toimivaa ratkaisua internetistä. Ainoaksi ongelmaksi jäi saada infolaatikon arvo dynaamiseksi, sillä se ei toiminutkaan suoraan niin kuin ajattelin. Pienen selvittelyn jälkeen löysin oikean tavan välittää muuttujan arvo infolaatikkoon.

Koska päivän tavoite tuli täyteen jo lounasaikaan mennessä ja en käyttöliittymää testatessa ollut törmännyt uusiin ohjelmavirheisiin, päätin alkaa tutustumaan Angular-sovelluksen automaattitesteihin. Käytin iltapäivän erinäisten ohjeiden ja esimerkkien lukemiseen ja yritin saada rakennettua edes yhtä testiä siinä onnistumatta. Ne esimerkit ja ohjeet, mitä onnistuin löytämään, käsittelivät paljon pienempiä sovelluksia. Aion huomenna tehdä jonkin pienen sovelluksen, jos vain löydän aikaa siihen, johon kokeilen kirjoittaa muutaman testin, jos sitä kautta ymmärtäisin koko konseptin paremmin ja osaisin tehdä testejä meidänkin sovellukseen.

Perjantai 1.6.2018

Halusin asettaa viimeisen seurantapäivän kunniaksi mahtipontisen tavoitteen päivälle, mutta en valitettavasti keksinyt mitään suurempaa tekemistä, joten jouduin tyytymään yksinkertaisempaan tavoitteeseen, eli tehdä backendiin uusi rajapinta. Tämän rajapinnan tarkoitus oli oikeastaan identtinen, kuin aikaisemmin tekemässäni rajapinnassa, jota käytetään PDF-tiedoston luomiseen. Tässäkin rajapinnassa luodaan PDF-tiedosto, mutta siihen käytettävä data on tulevaisuudessa todennäköisesti hyvin erinäköistä, sillä tarkoituksena on tehdä PDF hakustatistiikkadatasta.

Koska pystyin hyödyntämään jo olemassa olevaa metodia PDF-tiedoston luomiseen ja rajapinnan toteutukseen oli mahdollista ottaa mallia jo olemassa olevasta rajapinnasta, ei työ ollut kovin haastava. Huomasin rajapintaa tehdessäni, että olisin voinut käyttää jopa aikaisemmin tekemääni yhtä wrapper-luokkaa hyödykseni, mutta päätin olla tekemättä niin, koska statistiikkaosiota ei ole vielä edistetty, voi PDF:n luomiseen käytettävä data muuttua vielä paljon. Jos data muuttuisi, joutuisin joka tapauksessa tekemään uudet Java-luokat sitä varten, joten tein sekä Java-luokan ja sille oman wrapper-luokan, joita voitaisiin tarvittaessa muuttaa ilman, että muutokset vaikuttaisivat sovelluksen muihin toimintoihin.

Saatuani rajapinnan valmiiksi, keskityin sovelluksen käyttöliittymän hiomiseen ja toiminnallisuuden parantamiseen.

Viikkoanalyysi

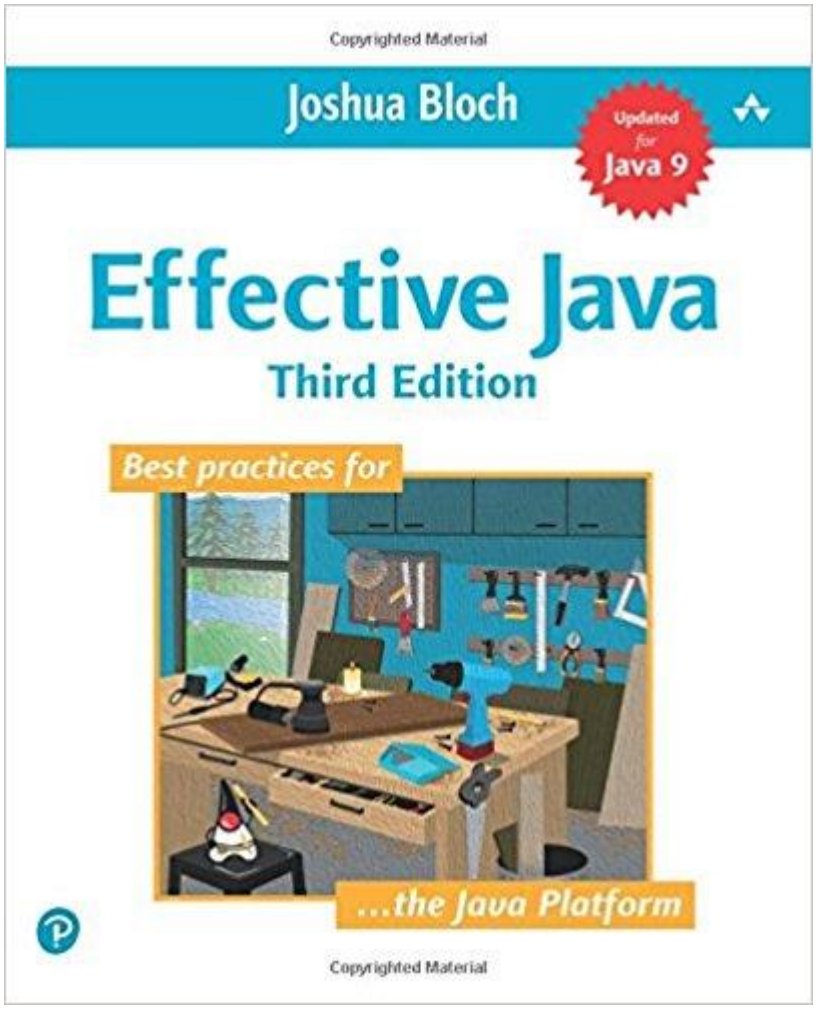
Seurantajakson viimeinen viikko osoittautui mielenkiintoiseksi ja melko tuotteliaaksi. Sain tehtyä paljon refaktorointia ja sitä kautta sovelluksen lähdekoodin selkeämmäksi ja helposti lähestyttävämmäksi. Tietysti tätä on hankala arvioida objektiivisesti, kun itse on ollut tekemässä sovellusta alusta asti, mutta uskon onnistuneeni siinä asiassa joka tapauksessa. Olen tyytyväinen siihen, että sain onnistuneesti muutettua sovelluksen käyttämään uudelleen URL-osoitteen mukana kulkevia hakuparametreja, koska pelkästään se muutos yksinään yksinkertaisti sovelluksen toimintalogiikkaa huomattavasti. Viikon aikana sain lisättyä myös pari uutta ominaisuutta, mikä onkin mukavampaa, kuin lähdekoodin refaktorointi tai virheenselvittely. Viikon kohokohdaksi muodostui kuitenkin se hetki, kun asiakas tiedusteli halukkuuttani jatkaa heidän projektien parissa. Aloittelevalle sovelluskehittäjälle tällainen merkitsee todella paljon.

Osaamiseni viikon aikana kehittyi ehkä lähinnä ainoastaan muiden kirjoittaman ohjelmakoodin lukemisessa. Tämä ei kuitenkaan ole ollenkaan huono asia, koska se on erittäin tärkeä osa ohjelmistokehittäjän arkea. Olen varma siitä, että lähestulkoon jokainen ohjelmistokehittäjä käyttää urallaan enemmän aikaa muiden kirjoittaman koodin lukemiseen kuin koodin kirjoittamiseen.

Käytin viikolla myös hieman aikaa täysin uuden asian opiskeluun eli yksikkötestaukseen. En vielä onnistunut sisäistämään asiaa kunnolla, joten tulen varmasti käyttämään vapaa-aikaani sen asian opetteluun. Tässä ammatissa uusien asioiden jatkuva opiskelu on vaatimus ja ainakaan minulla työaika ei riitä siihen, joten joudun käyttämään vapaa-aikaani opiskeluun. En koe tätä huonona asiana, mutta se on joka tapauksessa stressaavaa, koska vapaa-aikani on melko vähissä ilman sitäkin. Jo ennen kuin aloitin tässä työssä, heräsin joka aamu tuntia aikaisemmin, jotta pystyin opiskelemaan ohjelmointia ja edistääkseni omia ohjelmointiprojektejani. Teen edelleen samaa, mutta sen lisäksi käytän entistä enemmän omaa aikaa uusien asioiden opiskeluun, joten väistämättä välillä tuntuu, että elämäni on pelkkää ohjelmointia. Toistaiseksi se ei haittaa, mutta pidemmän päälle se voi käydä raskaaksi, mutta luotan siihen, että mitä paremmaksi osaamiseni kehittyy, sitä vähemmän vapaa-aikaa minun tarvitsee käyttää uusien asioiden opettelemiseen.

Tapoja uusien asioiden opettelemiseen on monia eivätkä kaikki tavat sovi jokaiselle. Kuten Sonmez listaa blogissaan, voi uusia asioita opetella esimerkiksi lukemalla blogeja tai kirjoja, osallistumalla erilaisiin tapahtumiin tai seuraamalla alan uutisointia (Sonmez 2017). Itse olen näistä tavoista ottanut omakseni viime aikoina kirjojen lukemisen ja yritän löytää jokaisesta päivästä vähintään puoli tuntia lukemiseen. Kirja, jota tällä hetkellä luen, on

Joshua Blochin kirjoittama Effective Java (kuvio 11), jonka toivon auttavan minua käyttämään Java-ohjelmointikieltä paremmin. Kirjat ovat hyvä tietolähde, koska yleensä ne sisältävät myös paljon teoriaa ja kirjoissa voidaan asioihin paneutua syvällisemmin kuin esimerkiksi videoissa tai blogikirjoituksissa. Vaikka pidänkin lukemisesta, on tämä opiskelutapa kuitenkin hidas, joten siksi suosinkin yleensä videomuotoisia tutoriaaleja, koska niiden katsomiseen ei mene niin paljoa aikaa.



Kuvio 11. Effective Java kirjan kansi (Google 2018b)

Ehdottomasti tärkein asia uusien asioiden opettelemisessa ja taitojen ylläpitämisessä on ohjelmoida mahdollisimman paljon ja ideaalitilanteessa joka päivä. Idea tämän takana on yksinkertainen, sillä mitä enemmän käytät aikaa ohjelmoimiseen, sitä epätodennäköisempää on, että ohjelmointitaidot ruostuisivat. Tähän olen itse pyrkinyt ja tämän vuoden alusta katsottuna ei ole ollut yhtäkään päivää, ettenkö olisi käyttänyt vapaa-aikaani vähintään 30 minuuttia ohjelmoimiseen. Tämä on itselleni paras tapa opetella uusia asioita ja ylläpitää nykyisiä taitoja. Avain tähän on kuitenkin se, että täytyisi olla jokin oma sivuprojekti, jota voi tehdä vapaa-aikana.

Ajatus päivittäiseen ohjelmoimiseen heräsi, kun törmäsin internetissä 100 päivää koodausta – nimiseen haasteeseen, jossa oli tarkoitus koodata jokaisena päivänä sadan päivän ajan. Haasteessa on kaksi sääntöä, joista ensimmäinen on, että joka päivä on käytettävä vähintään tunti ohjelmoimiseen ja toinen sääntö on, että oma edistyminen pitäisi twiitata joka päivä #100DaysOfCode – tunnisteella (#100DaysOfCode 2018). Haasteen aloittamisen jälkeen löysin hyvän kirjoituksen eduista, joita voi saavuttaa ohjelmoimalla joka päivä. Resig listaa kirjoituksessaan eduiksi muun muassa, että ohjelmoiminen tuli tavaksi, pelko siitä, että omat projektit eivät edisty hävisi, viikonlopuista tuli stressittömämpiä, koska projektit edistyivät jo viikon aikana, jolloin stressi niiden edistämiseen viikonlopuilta hävisi. Mielestäni tärkein asia, jonka hän huomasi oli se, että hän ajatteli ohjelmointia koko ajan, mikä johti siihen, että eteen tulevat ongelmat ratkesivat jo ennen kuin hän alkoi kirjoittamaan koodia (Resig 2014).

Olen itse huomannut samoja asioita tapahtuvat osittain itsellenikin, mutta pelko omien projektien edistämisestä on vielä olemassa. Toivon, että tämä häviää ajan kanssa. Kuten jo mainitsinkin aiemmassa kappaleessa, että tärkein asia mielestäni koko prosessissa oli se, että hän ajatteli ohjelmointia koko ajan. Tärkeimmäksi asiaksi tämän näen siksi, että se säästää aikaa pitkällä aikavälillä paljon, koska ohjelmoinnin aikana ei tarvitse käyttää niin paljon aikaa ongelmien ratkomiseen vaan voi keskittyä kirjoittamaan koodia. Lisäksi, kun ohjelmointia ajattelee niin sanotusti koko ajan, on koodin kirjoittaminen helpompi aloittaa, koska on tavallaan jo valmiiksi oikeassa mielentilassa siihen.

Uusien asioiden opiskelussa olisi hyvä olla myös suunnitelma, jotta opiskelu ei ole päätöntä juoksemista asiasta toiseen. Itse sorrun tähän monesti eli hypin asioiden välillä ilman sen tarkempaa suunnitelmaa, että mitä kannattaisi opiskella ja miksi. Sen sijaan luen vähän joka asiasta päivän tuntemuksien mukaan. Mitään hyötyä tästä ei varsinaisesti ole, koska kun asiaan ei keskity kunnolla tai ei käytä tarpeeksi aikaa asian sisäistämiseen, ei siitä jää mieleen juuri mitään. Olenkin viime aikoina pyrkinyt olemaan johdonmukaisempi uusien asioiden opettelussa ja keskittyä niihin asioihin, joita minun pitää nyt osata ja mitkä ovat tärkeimpiä työni menestyksekkäässä hoitamisessa. Tämä johtuu osittain ehkä siitä stressistä, jonka tämä työ aiheuttaa. Haluan kyetä hoitamaan työni kunnialla ja se tarkoittaa sitä, että minulla on jatkuvasti olo, että pitäisi opetella ja osata kaikkea.

4 Pohdinta ja päätelmät

Menneet kahdeksan viikkoa ovat olleet todella opettavaista aikaa. Olen saanut olla epämukavuusalueella koko tämän ajan ja sitä kautta haastaa itseäni jatkuvasti. Uusia asioita on tullut valtavasti eteen – paljon enemmän kuin aluksi edes kuvittelin. Mukaan on mahtunut niin onnistumisia kuin ikäviäkin asioita, stressiä ja kiireetöntä tekemistä. Jos viikot eivät jotain ole olleet, niin tylsiä. Ehkä parasta on ollut jatkuva uusien asioiden opiskelu, vaikka se on melko stressaavaa välillä.

Kun aloin tekemään opinnäytetyötä päiväkirjan muodossa, päätin jo silloin, että kirjoitan päivän tapahtumat muistiin sitä mukaa, kun ne tapahtuvat ja heti työpäivän jälkeen kirjoitan sen päivän auki tähän työhön ja viikkoanalyysin aina perjantaina. Tämä lähestymistapa siksi, että päivä on vielä tuoreessa muistissa ja perjantaina on helppo kirjoittaa viikkoanalyysi heti päivätekstin kirjoittamisen jälkeen, koska olisin jo valmiiksi asennoitunut tekstin kirjoittamiseen. Näin jälkikäteen on helppo todeta, että onneksi tein niin ja pysyin päätöksessäni. Luonnollisestikaan aina ei olisi ollut mielenkiintoa tai välttämättä edes aikaa alkaa kirjoittamaan päiväkirjaa, mutta tiesin, että kun ensimmäisen kerran annan itselleni periksi, on toinen kerta paljon helpompi.

Päiväkirjaa pitäessäni huomasin, että koska kirjoittaessa pohdin asioita uudemman kerran, vahvisti se osaamistani ihan uudella tavalla ja muistijälki päivästä vahvistui. Ehkä päiväkirjan pitäminen ei olisi ollenkaan huono ratkaisu tulevaisuudessakaan oman osaamisen kehittämisessä ja ylläpitämisessä. Normaalitilanteessa kuitenkin töistä tullessaan yleensä pyrkii keskittymään ihan muihin asioihin ja työpäivän aikana käsitellyt asiat hautautuvat muistin uumeniin.

Itse kirjoittamisessa suurin haaste oli ehdottomasti se, että jouduin kuvailemaan kaiken tekemäni hyvin yleisellä tasolla. Olisi ollut mukavaa, jos olisin voinut joskus vaikka laittaa kuvan sovelluksesta havainnollistamaan lukijalle paremmin, että mitä olen tehnyt ja miksi. Toinen ongelmakohta oli yrittää kirjoittaa päivästä tekstiä silloin, kun mitään ei ole oikein tapahtunut. Välillä oli sellainen olo, että jouduin oikein keksimällä keksimään jotain, mistä kirjoittaa tähän, koska työpäivät eivät aina ole kovin tapahtumarikkaita. Tämä heijastui osittain myös viikkoanalyysien kirjoittamiseen – oli vaikea kirjoittaa tarpeeksi pitkästi viikosta, jos viikon tapahtumat mahtuivat yhteen kappaleeseen. Tässä pelastukseksi osoittautui kuitenkin tietolähteiden hyödyntäminen ja tietolähteitä lukiessani kehitin myös samalla osaamistani ja vahvistin jo olemassa olevaa tietopohjaa.

Mainitsin jo seurantaviikon 18. viikkoanalyysissä, että lähtötilanteeni Angular-ohjelmistokehyksen kanssa ei ollut kovin hääppöinen. Olin tehnyt jotain pientä harrastelumielessä ja suoritin yhden kurssin Udemyssä ennen projektin alkua. Nyt kun katson taaksepäin ja vertaan alkutilannetta nykyhetkeen, osaamiseni on aivan eri tasolla. Kehitystä on tapahtunut hurjasti aina Angular applikaation arkkitehtuurista sen jokapäiväiseen käyttöön työelämässä. Vaikka en edelleenkään ole osaamistasoltani mikään guru sen suhteen, voin kuitenkin luottavaisin mielin hypätä seuraavaan projektiin, missä Angularia olisi tarkoitus käyttää. Se, että koska se tapahtuu, on luonnollisesti vielä hämärän peitossa, koska olen nykyisissä projekteissa aina vuoden loppuun saakka. Näiden jälkeisessä mahdollisessa seuraavassa projektissa missä olen mukana, voidaan käyttää eri teknologioita ja projektin kesto voi olla jopa vuoden tai kahden mittainen. Seurantajakson puolivälissä muistan olleeni vielä vähän epävarma, mitä tulee käyttöliittymän ohjelmointiin Angularilla, mutta se epävarmuus on haihtunut tyystin seurantajakson jälkimmäisellä puolikkaalla.

Näin jälkikäteen voisin sanoa, että olin mukana harmillisen vähän palvelinpuolen ohjelmoinnissa, koska se on edelleen se suurin kiinnostuksenkohteeni. Varmasti suurin syy vähyyteen oli se, että olen hyvin epävarma Javan ja etenkin Spring Bootin käytöstä, johtuen kokemuksen puutteesta. Mutta kuinka kokemusta tulee, jos ei uskalla ryhtyä toimeen? Projekti ei mennyt ihan hukkaan kuitenkaan, sillä osaamistasoni Spring Bootissa on joka tapauksessa todella paljon parempi nyt kuin ennen projektin alkua. Se oli kyllä odotettavissakin, koska osaamista ei käytännössä ollut ollenkaan. Seurantajakson aikana ilmaisinkin työnantajalleni tahtoni erikoistua Javaan ja Spring ohjelmistokehykseen ja toivoin saavani mentorin, joka voisi opastaa minua matkalla vakavasti otettavaksi Java-kehittäjäksi.

Osaamiseni kehittyi myös monessa muussa asiassa, mitkä eivät suoranaisesti liity itse ohjelmakoodin kirjoittamiseen. Näiden kuluneen kahdeksan viikon aikana olen oppinut lukemaan ja ennen kaikkea ymmärtämään virallista dokumentaatiota todella paljon paremmin. Ennen saatoin lukea dokumentaatiota ja todeta, että en ymmärrä mitään ja sen jälkeen etsinyt valmiita esimerkkejä internetistä ongelmaan liittyen. Nykyään kykenen paljon paremmin soveltamaan dokumentaation tarjoamia esimerkkejä eikä minun tarvitse etsiä valmiita ratkaisuja lähellekään niin usein kuin ennen. Tähän asiaan olen erittäin tyytyväinen. Toinen asia, missä olen kehittynyt ja mikä ei liity suoraan ohjelmakoodin tuottamiseen, on uusien asioiden opettelu. Koska ymmärrykseni ohjelmoinnista yleisellä tasolla on noussut kohisten, on se helpottanut uusien asioiden opiskelua huomattavasti. On paljon helpompi imeä itseensä uutta informaatiota kun perusta on vakaammalla pohjalla. Vaikka en vielä olekaan kollegoideni tasolla siinä asiassa, on ero seurantajaksoa edeltä-

vään aikaan todella suuri. Tämän seurantajakson aikana olen myös oppinut antamaan itselleni anteeksi sen, että en ole niin hyvä kuin kollegani – työkokemusero meidän välillä on kuitenkin liian suuri kurottavaksi kiinni muutaman kuukauden mittaisen projektin aikana.

Kokeneiden kollegoiden työskentelyä ja vuoropuhelua oli nautinto seurata. Opin todella paljon pelkästään kuuntelemalla heitä, kun he ratkaisivat eritasoisia ja -kokoisia ongelmia. Ennen kuin aloitin työt ohjelmistokehittäjänä ja tein ohjelmointia vain omalla ajalla harrastuksena, olin tottunut siihen, että kaikki ongelmat, joita vastaan tuli, piti ratkaista yksin. Silloin tärkein työkalu oli internetsivu StackOverflow, joka on varmasti kullan arvoinen sivusto suurimmalle osalle ohjelmistokehittäjistä. Olikin siis mukavaa ja tervetullutta vaihtelua, kun vierellä oli kokeneita työkavereita, joilta pystyi kysymään apua ja lähes poikkeuksetta vastaus löytyi. Jos ei löytynyt, etsimme sen yhdessä.

Seurantajakson aikana ehkä eniten yllätyin siitä, että kuinka nopeasti osaava ohjelmistokehittäjä kykenee opettelemaan uusia asioita. Ulkopuolisin silmin prosessi näytti siltä, että dokumentaation läpi lukeminen kerran riitti, jonka jälkeen pienen testailun jälkeen hän oli valmis hyödyntämään tätä uutta teknologiaa.

Tässä projektissa nousi todella usein esiin yksi asia, joka oli sovellusarkkitehtuuri ja sen tärkeys. Eteen tulleet ongelmat siihen liittyen herättivät minut ajattelemaan asiaa. Ennen seurantajakson alkua en juurikaan murehtinut ohjelman kokonaisarkkitehtuurista, vaan ryhdyin aina kirjoittamaan koodia ilman, että oikeastaan pysähdyin miettimään koko asiaa. Mentaliteettini silloin oli, että muutetaan, jos tarve muutokselle ilmenee. Nyt en voisi kuvitellakaan ryhtyväni työhön ilman, että ensin oikeasti ajattelen kokonaiskuvaa ja arkkitehtuuria. Luonnollisestikaan arkkitehtuuri ei ole niin tärkeä, kun puhutaan pienistä sovelluksista, mutta mitä suurempi sovellus, sitä tärkeämpi se on. Seurantajaksolla sain seurata aitiopaikalta, mihin väärin tehdyt valinnat arkkitehtuurin suhteen voivat johtaa ja kuinka hankalaa ne ovat oikeasti korjata työelämässä vähänkään suuremmissa projekteissa. Arkkitehtuurin muuttaminen voi tarkoittaa todella suurta määrää työtä ja sitä kautta rahan menoa ja se voi ja todennäköisesti myös hidastaa jo meneillään olevia muita sovelluskehitysprojekteja. Tämän oivaltaminen oli minulle äärimmäisen tärkeää.

Kuluneen kahdeksan viikon aikana kokonaan uusia asioita minulle tuli vastaan melkoinen määrä. Niistä isoin kokonaisuus oli kyllä ehdottomasti sovelluskehityksen tekeminen työkseen verrattuna harrasteluun ja omiin projekteihin, joita olin tehnyt ennen tätä työtä. Ehdin olla kuukauden verran tässä projektissa mukana ennen kuin seurantajakso alkoi, mutta varsinainen sovelluskehitys alkoi mielestäni vasta sitten kun kollegani pääsivät mukaan

tähän projektiin. Ennen sitä työskentely oli suurimmalta osin verrattavissa omaan harrastusprojektiin, koska tein töitä enimmäkseen yksin, ei ollut varsinaisia deadlineja, kukaan ei ollut kommentoimassa koodiani ja minun ei tarvinnut ottaa ketään muuta huomioon ohjelmakoodia tehdessä. Kaikki tämä muuttui seurantajakson aikana, kun ensimmäinen työkaverini pääsi kunnolla tekemään töitä tämän projektin parissa. Sain palautetta tekemistäni ratkaisuista, parannusehdotuksia ja pääsin seuraamaan kokeneemman kehittäjän tapaa ajatella ja ratkaista ongelmia sekä lukemaan hänen tuottamaa koodia. Alkuun tämä oli melko stressaavaa, että joku muukin kuin minä, lukisi kirjoittamaani koodia ja näkisi tekemäni ratkaisut, mutta jo seurantajakson puolivälin tienoilla stressi oli hävinnyt miltei kokonaan. Sain pääosin hyvää palautetta siitä, mitä olin tehnyt ja tekemiini virheisiin suhtauduttiin ymmärtäväisesti.

Toinen suurempi kokonaisuus, mikä oli minulle kokonaan uutta työelämässä, oli työskentely asiakkaan tiloissa. Tätä ennen olin tehnyt töitä vain oman työnantajani tiloissa, joten muutos oli aika suuri. Oli tärkeää pitää mielessä, että edustan työnantajaani ja heidän maineensa on pelissä – ei minun. Onnekseni pääsin mukaan projektiin, missä pystyin olemaan oma itseni. Lisäksi minua kohdeltiin kuin omaa työntekijää, mikä oli positiivista. Vaikka kaikki menikin hyvin ja meininki oli pääosin rentoa, en kyennyt hengittämään ihan täysin vapaasti. Tämä oli kuitenkin pieni ongelma, jos sitä voi sellaiseksi edes sanoa ja kaiken lisäksi ongelma oli lähinnä vain itsestäni kiinni.

Kirjoitusprosessin aikana minulle heräsi sellainen ajatus, että olisi mielenkiintoista pitää vastaavanlainen seurantajakso parin vuoden päästä. Olisi mielenkiintoista päästä vertaamaan, että kuinka paljon asiat ovat muuttuneet vuoden tai kahden jälkeen. Tämä työ on mielestäni sellainen, että sitä ei voi kunnolla oppia harjoittelemalla vain kotona tekemällä omia projekteja, sillä niistä puuttuu yleensä työelämän vaatima laajuus ja monipuolisuus. Esimerkkinä sanottakoon, että en todennäköisesti olisi kovin äkkiä ryhtynyt luomaan PDF-tiedostoja HTML-sapluunoiden pohjalta.

Opinnäytetyön kirjoittaminen itsessään oli myös opettava prosessi. Tämän kirjoittaminen pakotti minut lukemaan enemmän dokumentaatioita ja tutkimaan eri tietolähteitä ja myös ilmaisemaan itseäni kirjallisesti laajemmin kuin muutamalla sanalla. Nyt minulle ei tuota minkäänlaisia ongelmia muuttaa asia, jonka olisin aiemmin ilmaissut parilla sanalla, useamman kappaleen pituiseksi tekstiksi. Olikin ilo huomata, että kehitystä tapahtui muuallakin kuin työtehtävän suoranaisesti vaatimisissa taidoissa. Mutta tärkeimmäksi asiaksi koen sen, että opinnäytetyön kirjoittaminen sai minut analysoimaan itseäni ja työskentelytapojani ja sitä kautta tajuamaan ne kohdat itsessäni, mitkä kaipaavat kehitystä.

Lähteet

#100DaysOfCode. 2018. Luettavissa: http://www.100daysofcode.com/. Luettu: 1.6.2018.

Bach, J. 2003. The Challenge of ”Good Enough” Software. Luettavissa: http://www.satisfice.com/articles/gooden2.pdf. Luettu: 25.5.2018.

Barard, M. 2017. 8 Tips to Reading Documentation: A Newbie’s Guide. Luettavissa: http://blog.techtalentsouth.com/8-tips-to-reading-documentation-a-newbies-guide. Luettu: 24.5.2018.

Bloomberg, J. 2015. Has Agile Outlived Its Usefulness?. Luettavissa: https://www.forbes.com/sites/jasonbloomberg/2015/12/11/has-agile-outlived-its-usefulness/2/#2ee205366b07. Luettu: 25.5.2018.

DeRegnaucourt , J. 2018. n-cube. Luettavissa: https://github.com/jdereg/n-cube. Luettu: 25.4.2018.

Docker 2018a. Docker Docs. Luettavissa: https://docs.docker.com. Luettu: 26.4.2018.

Docker 2018b. What is Docker. Luettavissa: https://www.docker.com/what-docker. Luettu: 26.5.2018.

Eng, R. K. 2017. What is Programmer Burnout?. Luettavissa: https://hackernoon.com/what-is-programmer-burnout-651aa48984ef. Luettu: 18.5.2018

Fellner, M. & Persa. D. 2017. The Recipe For Scalable Frontends (Zalando). Luettavissa: https://www.youtube.com/watch?v=m32EdvitXy4. Luettu: 11.5.2018

Git 2018. Getting Started – A Short History of Git. Luettavissa: https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git. Luettu: 25.5.2018.

Google 2018a. Angular Docs. Luettavissa: https://angular.io/. Luettu: 20.4.2018.

Google 2018b. Effective Java. Luettavissa: https://books.google.fi/books?id=ka2VUBqHiWkC&printsec=frontcover&hl=fi&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false. Luettu: 1.6.2018.

Kasireddy, P. 2016. A Beginner-Friendly Introduction to Containers, VMs and Docker. Luettavissa: https://medium.freecodecamp.org/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b. Luettu: 27.4.2018.

Kothari, A. 2018. Why has Vue.js become so popular? Luettavissa: https://hub.packtpub.com/why-has-vuejs-become-so-popular/. Luettu: 26.5.2018

McEwen, M. 2017. I just don’t want to be a software developer anymore. Luettavissa: https://medium.com/@melissamcewen/i-just-dont-want-to-be-a-software-developer-anymore-a371422069a1. Luettu: 3.5.2018.

Project Mosaic 2016. Project Mosaic | Microservices for the Frontend. Luettavissa: http://www.mosaic9.org. Luettu: 11.5.2018.

ReactiveX 2018. Subject. Luettavissa: http://reactivex.io/documentation/subject.html. Luettu: 18.5.2018.

Resig, J. 2014. Write Code Every Day. Luettavissa: https://johnresig.com/blog/write-code-every-day/. Luettu: 1.6.2018.

State Of JavaScript 2017. Front-end Frameworks – Results. Luettavissa: https://stateofjs.com/2017/front-end/results. Luettu: 26.5.2018.

Somnez, J. 2017. Keeping Your Skills Up to Date as a Software Developer. Luettavissa: https://simpleprogrammer.com/developer-skills-updated/. Luettu: 1.6.2018.

Thymeleaf 2018. Tutorial: Using Thymeleaf. Luettavissa: https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html. Luettu: 25.5.2018.

Trautman 2015. Why Learning to Code is So Damn Hard. Luettavissa: https://www.thinkful.com/blog/why-learning-to-code-is-so-damn-hard/. Luettu: 28.5.2018.

Wikipedia 2009. HyperCard. Luettavissa: https://fi.wikipedia.org/wiki/HyperCard. Luettu: 25.5.2018.